

---

## **IMPULSE RADAR ANALYSIS**

**Robert J. Torres**

**December 1995**

**Final Report**

---

**APPROVED FOR PUBLIC RELEASE; DISTRIBUTION IS UNLIMITED.**

---

19960501 162



**PHILLIPS LABORATORY**  
**Advanced Weapons and Survivability Directorate**  
**AIR FORCE MATERIEL COMMAND**  
**KIRTLAND AIR FORCE BASE, NM 87117-5776**

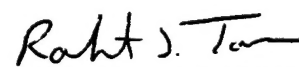
Using Government drawings, specifications, or other data included in this document for any purpose other than Government procurement does not in any way obligate the U.S. Government. The fact that the Government formulated or supplied the drawings, specifications, or other data, does not license the holder or any other person or corporation; or convey any rights or permission to manufacture, use, or sell any patented invention that may relate to them.

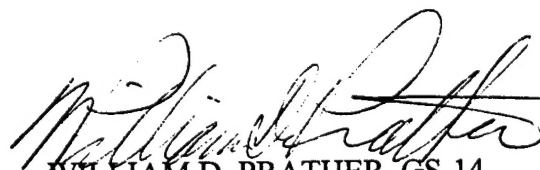
This report has been reviewed by the Public Affairs Office and is releasable to the National Technical Information Service (NTIS). At NTIS, it will be available to the general public, including foreign nationals.

If you change your address, wish to be removed from this mailing list, or your organization no longer employs the addressee, please notify PL/WSQW, 3550 Aberdeen Ave SE, Kirtland AFB, NM 87117-5776.

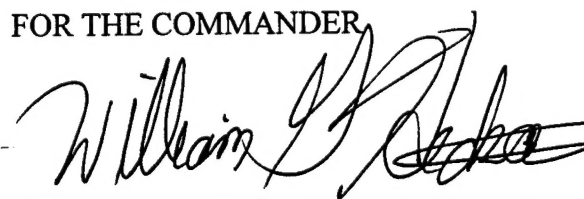
Do not return copies of this report unless contractual obligations or notice on a specific document requires its return.

This report has been approved for publication.

  
ROBERT J. TORRES  
Project Manager

  
WILLIAM D. PRATHER, GS-14  
Chief, Wideband Sources Branch

FOR THE COMMANDER

  
WILLIAM G. HECKATHORN, Col, USAF  
Director, Advanced Weapons and  
Survivability Directorate

# DRAFT SF 298

<b>1. Report Date (dd-mm-yy)</b> 15-12-95		<b>2. Report Type</b> Final		<b>3. Dates covered (from... to )</b> Jan 95 - Dec 95	
<b>4. Title &amp; subtitle</b> Impulse Radar Analysis				<b>5a. Contract or Grant #</b>	
				<b>5b. Program Element #</b> <div style="text-align: right;">62601F</div>	
<b>6. Author(s)</b> Robert J. Torres				<b>5c. Project #</b> <div style="text-align: right;">5797</div>	
				<b>5d. Task #</b> <div style="text-align: right;">AK</div>	
				<b>5e. Work Unit #</b> <div style="text-align: right;">07</div>	
<b>7. Performing Organization Name &amp; Address</b> Phillips Laboratory 3550 Aberdeen Avenue, SE Kirtland AFB, NM 87117-5776				<b>8. Performing Organization Report #</b>  PL-TR-96-1025	
<b>9. Sponsoring/Monitoring Agency Name &amp; Address</b>				<b>10. Monitor Acronym</b>	
				<b>11. Monitor Report #</b>	
<b>12. Distribution/Availability Statement</b> Approved for public release; distribution is unlimited.					
<b>13. Supplementary Notes</b>					
<b>14. Abstract</b> Impulse radars are of interest because of their range resolution. As such they have been widely used in radar imaging and remote sensing for target signature analysis and identification/classification. A comparison of an impulse waveform and a pulsed continuous waveform (CW) is analyzed. For each waveform, a Finite Difference Time Domain (FDTD) algorithm is used to obtain the radar cross section for three perfect conducting objects. It is shown that by having a larger frequency bandwidth, the impulse waveform is able to excite target resonances that cannot be excited by the pulsed CW waveform. Therefore, the impulse waveform provides greater resolution of an object.					
<b>15. Subject Terms</b> Impulse Radar, Radar Cross Section, Finite Difference Time Domain, Pulsed CW Radar, Perfect Conducting Objects.					
<b>Security Classification of</b>			<b>19. Limitation of Abstract</b>	<b>20. # of Pages</b>	<b>21. Responsible Person (Name and Telephone #)</b>
<b>16. Report</b> Unclassified	<b>17. Abstract</b> Unclassified	<b>18. This Page</b> Unclassified	Unlimited	196	Robert J. Torres (505) 846-0296

# **Impulse Radar Analysis**

by

Robert J. Torres

BS, Electrical Engineering, University Of New Mexico, 1986

THESIS

Submitted in Partial Fulfillment of the

Requirements for the Degree of

Master of Science in Electrical Engineering

The University of New Mexico

Albuquerque, New Mexico

December 1995

# Impulse Radar Analysis

by

Robert J. Torres

Master of Science in Electrical Engineering, University of New Mexico, 1995

Bachelor of Science in Electrical Engineering, University of New Mexico, 1986

## ABSTRACT

Impulse radars are of interest because of their range resolution. As such they have been widely used in radar imaging and remote sensing for target signature analysis and identification/classification. A comparison of an impulse waveform and a pulsed continuous waveform (CW) is analyzed. For each waveform, a Finite Difference Time Domain (FDTD) algorithm is used to obtain the radar cross section for three perfect conducting objects. It is shown that by having a larger frequency bandwidth, the impulse waveform is able to excite target resonances that cannot be excited by the pulsed CW waveform. Therefore, the impulse waveform provides greater resolution of an object.

## Table of Contents

	Page
1. Introduction	1
2. Radar Range Equation	3
3. Backscattering From an Object	8
4. Description of Waveforms	19
5. Finite Difference Time Domain Analysis	28
6. Description of the Targets	36
7. Description of Algorithm Runs	40
8. Analysis of Results	42
9. Conclusions	62
10. References	66
11. Appendix	68

## List of Figures

Figure	Title	Page
1.	Graph of Impulse Time Domain Waveform.	20
2.	Graph of Impulse Frequency Domain Waveform.	21
3.	Graph of Pulsed CW Time Domain Waveform.	22
4.	Graph of Pulsed CW Frequency Domain Waveform.	23
5.	Graph of Gaussian Time Domain Waveform.	26
6.	Graph of Gaussian Frequency Domain Waveform.	27
7.	Location of the Six-Field Evaluation Points in a Typical Yee Cell.	35
8.	Perfect Conducting Sphere. Radius = 19.5 Cells. Cell Size = 0.014m. 50 x 50 x 50 Cell Mesh.	37
9.	Perfect Conducting Square Plate. 30 x 30 Cells. 60 x 60 x 60 Cell Mesh.	38
10.	Complex Object - Perfect Conducting Elliptical Shape with a Cylinder Rod Through the Center.	39
11.	Radar Cross Section of Perfect Conducting Sphere obtained by Kunz and Luebbers [12] for Gaussian Input Waveform, 1000 Time Steps, Incident Angle = $22.5^{\circ}$ .	43
12.	Radar Cross Section of Perfect Conducting Sphere for Gaussian Input Waveform, 1000 Time Steps, Incident Angle = $22.5^{\circ}$ .	44
13.	Radar Cross Section of Perfect Conducting Sphere for Gaussian Input Waveform, 1000 Time Steps, Incident Angle = $0^{\circ}$ .	46
14.	Radar Cross Section of Perfect Conducting Sphere for Impulse Input Waveform, 5000 Time Steps, Incident Angle = $0^{\circ}$ .	47

15.	Radar Cross Section of Perfect Conducting Sphere for Pulsed CW Input Waveform, 5000 Time Steps, Incident Angle = $0^0$ .	48
16.	Radar Cross Section of Perfect Conducting Square Plate for Gaussian Input Waveform, 5000 Time Steps, Incident Angle = $0^0$ .	49
17.	Radar Cross Section of Perfect Conducting Square Plate for Impulse Input Waveform, 5000 Time Steps, Incident Angle = $0^0$ .	51
18.	Radar Cross Section of Perfect Conducting Square Plate for Pulsed CW Input Waveform, 5000 Time Steps, Incident Angle = $0^0$ .	52
19.	Radar Cross Section of Perfect Conducting Square Plate for Impulse Input Waveform, Horizontal Polarization, Incident Angle = $22.5^0$ .	53
20.	Radar Cross Section of Perfect Conducting Square Plate for Impulse Input Waveform, Vertical Polarization, Incident Angle = $22.5^0$ .	54
21.	Radar Cross Section of Perfect Conducting Square Plate for Pulsed CW Input Waveform, Horizontal Polarization, Incident Angle = $22.5^0$ .	56
22.	Radar Cross Section of Perfect Conducting Square Plate for Pulsed CW Input Waveform, Vertical Polarization, Incident Angle = $22.5^0$ .	57
23.	Radar Cross Section of Perfect Conducting Ellipsoid with a Cylinder for Impulse Input Waveform, Horizontal Polarization, Incident Angle = $22.5^0$ .	58
24.	Radar Cross Section of Perfect Conducting Ellipsoid with a Cylinder for Impulse Input Waveform, Vertical Polarization, Incident Angle = $22.5^0$ .	59
25.	Radar Cross Section of Perfect Conducting Ellipsoid with a Cylinder for Pulsed CW Input Waveform, Horizontal Polarization, Incident Angle = $22.5^0$ .	60

- 26. Radar Cross Section of Perfect Conducting Ellipsoid 61  
with a Cylinder for Pulsed CW Input Waveform,  
Vertical Polarization, Incident Angle =  $22.5^{\circ}$ .
- 27. Example of Resolution for an Object Using (a) Pulsed 63  
CW and (b) Impulse Waveform.

## **Introduction**

This thesis concerns the signal processing of the waveforms that are received from the backscattering of an object under test. The amplitude and phase of the return signals are used to construct a representation of the object. Therefore, the radar cross section signature of the object is identified. A comparison of an ideal impulse waveform and a pulsed continuous waveform (CW) is analyzed. In addition, a Gaussian waveform is used as a baseline. Three objects are identified using the impulse waveform and a pulsed CW waveform, and a comparison between the two waveforms for the three objects is made. The first object was a simple perfect conducting sphere shaped object. The second object was a perfect conducting Square plate and the third object was an elliptical shape with a cylinder rod through the center. The angle of incidence of the waveforms and polarization of the waveforms is also evaluated.

Impulse radars are of interest because of their range resolution. As such, they have been widely used in radar imaging and remote sensing for target signature analysis and identification/classification [1, 2, 3]. Two characteristics of impulse signals provide advantages over the conventional narrow band waveforms. The first characteristic of an impulse waveform is that its low frequencies can penetrate ground and foliage and can also excite target resonances. This allows for detection of low radar concealed targets in foliage, or detection of low radar cross section targets [4]. The second important characteristic is the very narrow pulse width, of the order of nanoseconds or less, with which fine resolution and low clutter is possible. With reduced clutter one can significantly improve the target detectability in strong range distributed clutter.

Clutter is the backscattering from the ground. Especially for conventional narrow band radars, the ground also becomes a target that can be confused between a real target and a background surface return. The ground roughness diminishes the specularity, and with it the forward scattering. Also, the reflected power is diverted to directions other than the specular direction including the backward direction. To measure low radar cross sections, all background signals must be reduced to levels that do not induce unacceptable errors.

Background signal level reduction is one of the basic principles of electromagnetic scattering measurements. Also, the measurements of scattered fields must be made in the far field of the scattered target where the incident field at the target must be a good approximation to a plane wave. Another important principle is that the power relations in electromagnetic scattering are governed by the radar range equation. This equation defines the smallest radar cross section that can be measured with a specific error.

## 2. Radar Range Equation

The radar cross section  $\sigma$  is embodied by the radar range equation. The radar range equation is the basic relationship which permits the calculation of backscatter signal strength from measurable or known parameters of the radar transmitter, antenna, propagation path, and target. The basic radar equation can be derived in the following manner. If the energy from a radar transmitter is radiated isotropically and the hypothetical isotropic antenna of the radar is ideal, meaning that the gain is unity, the uniform distribution of radiant energy over the surface of a sphere at range  $R_T$  will produce a power density driven by

$$\text{power / unit area} = P_T / 4\pi R_T^2 \quad (1)$$

where  $P_T$  is the power transmitted (usually taken as the average radio frequency power during the pulse) and  $R_T$  is the range of transmission [5]. Now one assumes that the transmitting antenna has a directive gain  $G_T$ . This directive gain referred to an ideal isotropic radiator is the ratio of  $4\pi$  times the ratio of the power radiated per unit solid angle in the defined direction to the total power delivered to the antenna. A target at range  $R_T$  would be illuminated by a wave of radiant energy whose power density is

$$\text{power / unit area} = P_T G_T / 4\pi R_T^2. \quad (2)$$

If we once again assume that the power is reflected isotropically, the power density of the wave reflected from the target at a distance  $R_R$  from the target at the receiving aperture is

$$\frac{\text{reflected power}}{\text{unit area}} = \left( \frac{P_T G_T}{4\pi R_T^2} \right) \left( \frac{\sigma}{4\pi R_R^2} \right) \quad (3)$$

where  $R_R$  is the receiving range, and  $\sigma$  is the radar cross section of the target in units consistent with range [5]. The radar cross section does not only depend on the frequency but also the target aspect with respect to both the transmitter and the receiver. If the transmitter and the receiver are not collocated then  $\sigma$  is the bistatic radar cross section. Next the amount of radiant energy intercepted by the receiving antenna is the product of the energy power density and the effective aperture area of the receiving antenna. The effective aperture is defined as

$$A_e = G_R \lambda^2 / 4\pi \quad (4)$$

where  $\lambda$  is the wavelength of the transmitted frequency,  $\lambda^2/4\pi$  is the universal antenna constant, and  $G_R$  is the gain of the receiving antenna [5]. The power received can now be expressed as

$$\text{received signal power} = S = \frac{P_T G_T \sigma A_e}{(4\pi)^2 R_R^2 R_T^2} \quad (5)$$

If the effective receiving aperture is substituted into the above equation, the reflected power at the receiver becomes

$$S = \frac{P_T G_T G_R \lambda^2 \sigma}{(4\pi)^3 R_T^2 R_R^2} \quad (6)$$

Other assumptions can be made to modify the radar range equation. For example, the transmitter and receiver could be collocated and use the same antenna. Then the radar equation can be expressed as

$$S = \frac{P_T G^2 \lambda^2 \sigma}{(4\pi)^3 R^4} \quad (7)$$

The ability of a radar to detect the presence of a echo signal is fundamentally limited by noise. Likewise, noise is the factor that limits the accuracy with which the radar signals may be estimated. The parameters usually of interest in radar applications are the range, or time delay, the range rate, or Doppler velocity, and the angle of arrival. The precise value of the amplitude of the echo signal is usually not important, except in so far as it influences the signal-to-noise ratio (SNR). To simplify the analysis, it is assumed that the signal is large compared with the noise. This is a reasonable assumption since the SNR must be relatively large, if the detection decision is to be reliable.

The radar range equation can be rewritten in a different form for detection range estimates. A required minimum SNR can be defined based on required detection probability, target statistics, and radar characteristics for the simple case of detection of a target in receiver noise. If it is assumed that the receiver noise is constant, the minimum SNR defines the maximum detection range by defining a minimum level of received signal,  $S_{\min}$ , which can be tolerated. Therefore, the maximum detection range is given by

$$R_{\max} = \left[ P_t G^2 \lambda^2 \sigma / (4\pi)^3 S_{\min} \right]^{1/4}. \quad (8)$$

This equation shows that the maximum detection range in free space varies only as the fourth-root of the radar cross section. For example, to reduce the maximum detection range by 3 dB, a 12 dB reduction in radar cross section is required. The maximum detection range becomes complicated when trying to detect in a clutter environment, because clutter will typically exhibit a received power versus range dependency varying from  $R^{-3}$  to  $R^{-7}$  [6].

The radar detection problem is normally approached through the concepts of statistical decision theory, because it involves the detection of a random time-varying signal in a randomly varying background [6, 7]. Two hypotheses are usually used in the radar detection decision. The first hypothesis is that no target is present (noise only). The second hypothesis is that a target is present (target return plus noise).

The error constraint normally used in radar is to limit the numbers of times the second hypothesis is chosen when in fact no target is present. This is known as a false alarm. To prevent false alarms involves correctly placing a threshold that noise alone will cross, on the average, at an acceptable rate. In order to know where to place the threshold, the statistics of the receiver noise needs to be known.

The detected received signal can be limited from thermal noise power generated by the random thermal motion of conduction electrons in the input stages. Usually an optimal receiver for the detection of a pulse train is comprised of a narrow-band filter “matched” to the single-pulse width, followed by a synchronous detector and an integrator. The available thermal noise power after the filter is a function of the temperature  $T$  and the bandwidth  $B_n$  of the receiver, and is given by

$$N = FKT B_n \quad (9)$$

where  $F$  is the receiver noise figure and  $K$  is Boltzmann’s constant ( $= 1.3 \times 10^{-23} \text{ J}^0\text{K}$ ) [8]. Now the radar range equation can be written in terms of SNR as

$$\text{SNR}_p = \frac{P_T G^2 \lambda^2 \sigma}{(4\pi)^3 R^4 FKT B_n} \quad (10)$$

where  $\text{SNR}_p$  is the SNR when only one pulse is returned from the target. The target is usually illuminated for a relatively long period of time  $t_1$ , and the number of pulses that can be used is  $M$ , where

$$M = t_1 f_R \quad (11)$$

and  $f_R$  is the pulse repetition frequency [8].

For this analysis, a few assumptions will be made in using the standard radar range equation for an impulse radar. Although the radar range equation assumes a narrowband, it will still be used for the impulse waveform. Wideband and impulse radar range equations have been derived [9, 10]. However, these will not be used because the impulse waveform employed does not have a very large bandwidth. Another assumption is that the gain of the transmit antenna and the receive antenna will be assumed to have constant gain over the bandwidth of interest.

### 3. Backscattering From an Object

In analyzing an electromagnetic diffraction problem, the fields can be separated into near and far fields. The near field does not only consist of Fresnel diffraction but is also responsible for charges and currents induced on the surfaces of the scattering bodies. The far field is responsible for the cross section of scattering bodies and therefore will be analyzed in detail. The far field is defined as the region of the field of an antenna where the angular field distribution is essentially independent of the distance from the antenna. The far field region exists at distances greater than  $2D^2/\lambda$  from the antenna where  $\lambda$  is the wavelength and  $D$  is the antenna length. The outer boundary is infinity. Therefore the far field can correspond to the Fraunhofer field of physical optics.

The far field consist of an electric (**E**) and magnetic (**H**) fields that can be expressed as sums  $\mathbf{E}_i + \mathbf{E}_s$  and  $\mathbf{H}_i + \mathbf{H}_s$ , respectively, which are the incident (i) and scattered (s) electric and magnetic vectors.  $\mathbf{E}_s$  and  $\mathbf{H}_s$  are defined by

$$\mathbf{E}_s = \lim_{R \rightarrow \infty} (\mathbf{E} - \mathbf{E}_i) \quad (12)$$

and

$$\mathbf{H}_s = \lim_{R \rightarrow \infty} (\mathbf{H} - \mathbf{H}_i) \quad (13)$$

With time dependence  $\exp(j\omega t)$  suppressed, the  $\mathbf{E}_s$  and  $\mathbf{H}_s$  vectors can be expressed as

$$\mathbf{E}_s(\mathbf{R}) \sim \mathbf{U}(\mathbf{k}, \mathbf{R}_0) \frac{e^{jkR}}{R} \quad (14)$$

and

$$\mathbf{H}_s(\mathbf{R}) \sim \mathbf{V}(\mathbf{k}, \mathbf{R}_0) \frac{e^{jkR}}{R} \quad (15)$$

where the  $\mathbf{U}$  and  $\mathbf{V}$  vectors are called scattering amplitudes,  $\mathbf{k}$  is the propagation vector, and  $\mathbf{R}_0$  is the direction to the point of observation in the far field [11]. To get exact relations for the far field amplitudes, the far field can be represented in terms of near field parameters. First, the total electric and magnetic field vectors can be expressed in terms of the incident field,  $\mathbf{E}_i$  and  $\mathbf{H}_i$ , and integrals over the scattering body surface where the material of the body can be dielectric and of finite conductivity as

$$\mathbf{E}(\mathbf{R}) = \mathbf{E}_i + \frac{1}{4\pi} \int_s [j\omega\mu(\mathbf{n} \times \mathbf{H})G + (\mathbf{n} \times \mathbf{E}) \times \nabla G + (\mathbf{n} \cdot \mathbf{E})\nabla G] dS' \quad (16)$$

and

$$\mathbf{H}(\mathbf{R}) = \mathbf{H}_i + \frac{1}{4\pi} \int_s [(\mathbf{n} \times \mathbf{H}) \times \nabla G + (\mathbf{n} \cdot \mathbf{H})\nabla G - j\omega\epsilon(\mathbf{n} \times \mathbf{E})G] dS' \quad (17)$$

By approximating the function  $G$  in the limit of large  $\mathbf{R}$ , an integral representation for  $\mathbf{U}$  and  $\mathbf{V}$  can be derived. This involves making the approximation

$$\begin{aligned} |\mathbf{R} - \mathbf{R}'| &= \left[ (\mathbf{R} - \mathbf{R}')^2 \right]^{1/2} = (\mathbf{R} \cdot \mathbf{R} - 2\mathbf{R} \cdot \mathbf{R}' + \mathbf{R}' \cdot \mathbf{R}')^{1/2} \\ &= R \left( 1 - \frac{2\mathbf{R} \cdot \mathbf{R}'}{R^2} + \frac{\mathbf{R}'^2}{R^2} \right)^{1/2} \sim R - \frac{\mathbf{R} \cdot \mathbf{R}'}{R} \end{aligned} \quad (18)$$

and dropping all terms in the function  $G$  of order higher than  $1/R$  [11].  $\mathbf{R}$  is the radius vector to a point of observation and  $\mathbf{R}'$  is the radius vector to a point on the scattering body. The result is

$$\begin{aligned} \mathbf{U}(\mathbf{k}, \mathbf{R}_0) &= \frac{jk}{4\pi} \int_s \left[ \left( \frac{\mu}{\epsilon} \right)^{1/2} (\mathbf{n} \times \mathbf{H}) - (\mathbf{n} \times \mathbf{E}) \times \mathbf{R}_0 - (\mathbf{n} \cdot \mathbf{E})\mathbf{R}_0 \right] e^{-jk\mathbf{R}_0 \cdot \mathbf{R}'} dS' \end{aligned} \quad (19)$$

and

$$\begin{aligned} & \mathbf{V}(\mathbf{k}, \mathbf{R}_0) \\ &= \frac{-jk}{4\pi} \int_s \left[ \left( \frac{\epsilon}{\mu} \right)^{1/2} (\mathbf{n} \times \mathbf{E}) + (\mathbf{n} \times \mathbf{H}) \times \mathbf{R}_0 + (\mathbf{n} \cdot \mathbf{H}) \mathbf{R}_0 \right] e^{-jk\mathbf{R}_0 \cdot \mathbf{R}'} dS'. \end{aligned} \quad (20)$$

Even though the far field itself is an approximation, the relations for  $\mathbf{U}$  and  $\mathbf{V}$  may be regarded as exact relations for the far field amplitudes [11].

In the above equations, the individual terms are as follows:  $\mathbf{n} \times \mathbf{H}$  is the electric surface current,  $\mathbf{n} \times \mathbf{E}$  is the magnetic surface current,  $\mathbf{n} \cdot \mathbf{E}$  the electric surface charge, and  $\mathbf{n} \cdot \mathbf{H}$  the magnetic surface charge. If the scattering body is a perfect conductor, the magnetic surface currents and magnetic surface charges will vanish. Therefore, the relations for  $\mathbf{U}$  and  $\mathbf{V}$  reduce to

$$\mathbf{U}(\mathbf{k}, \mathbf{R}_0) = \frac{jk}{4\pi} \int_s \left[ \left( \frac{\mu}{\epsilon} \right)^{1/2} (\mathbf{n} \times \mathbf{H}) - (\mathbf{n} \cdot \mathbf{E}) \mathbf{R}_0 \right] e^{-jk\mathbf{R}_0 \cdot \mathbf{R}'} dS' \quad (21)$$

and

$$\mathbf{V}(\mathbf{k}, \mathbf{R}_0) = \frac{-jk}{4\pi} \int_s [(\mathbf{n} \times \mathbf{H}) \times \mathbf{R}_0] e^{-jk\mathbf{R}_0 \cdot \mathbf{R}'} dS' \quad (22)$$

in which only the electric surface current and charge appear [11].

The scattered electric and magnetic vectors are used in determining the radar cross section. The scattering from a body in the optical limit is described by tracing the rays of the incident field on and reflected field from the body according to Snell's Law of reflection [11]. The incident and reflected rays and the normal to the scattering body surface lie in the plane of incidence. Also the angle of incidence relative to the surface normal is equal to the angle of reflection. By applying the boundary conditions, the

initial values for the field components on the reflected ray system are determined. An additional requirement on the initial field is implied by the fact that the electric and magnetic field vectors must be orthogonal to the ray along which the field propagates. By using a ray system and boundary conditions one can solve a scattering problem for both a dielectric body and a perfect conductor. For a dielectric body one can obtain reflection and transmission coefficients relative to the interface by using the standard Fresnel formulas. The refracted and reflected fields must be matched at the body by treating the matching problem as if the waves were plane waves and the dielectric interface were an infinite plane. However, to obtain a practical answer in a scattering calculation, one should neglect all but finite number of reflections and refractions [11].

Using a perfect conductor for a scattering body, the total tangential electric vector vanishes on the body's surface. The boundary conditions that exist are

$$\mathbf{n} \times \mathbf{E}_s = -\mathbf{n} \times \mathbf{E}_i \quad (23)$$

and

$$\mathbf{n} \cdot \mathbf{H}_s = \mathbf{n} \cdot \mathbf{H}_i. \quad (24)$$

There is also a boundary condition on the incident phase  $\psi_i$  and the scattered phase  $\psi_s$  according to the law of reflection in the optical limit which is

$$\mathbf{n} \times \nabla \psi_i = \mathbf{n} \times \nabla \psi_s. \quad (25)$$

Equation (25) states that an incident ray makes the same angle with the normal to the surface as the corresponding reflected ray and that the two rays and the surface normal lie in the same plane of incidence. Therefore the following condition exists.

$$\mathbf{n} \cdot \nabla \psi_i = -\mathbf{n} \cdot \nabla \psi_s. \quad (26)$$

The tangential component of  $\mathbf{E}_s$  can be derived by taking the cross product of  $\mathbf{n}$  with Equation (23) which is

$$\mathbf{E}_s - (\mathbf{n} \cdot \mathbf{E}_s) \mathbf{n} = \mathbf{n} \times (\mathbf{n} \times \mathbf{E}_i). \quad (27)$$

The normal components of  $\mathbf{E}_s$  and  $\mathbf{E}_i$  can be shown to be identical by taking the dot product of  $\nabla\psi_s$  and  $\nabla\psi_i$  successively with Equation (27) and applying Equation (26); that is,

$$\mathbf{n} \cdot \mathbf{E}_i = \mathbf{n} \cdot \mathbf{E}_s. \quad (28)$$

A compact form of the boundary condition for  $\mathbf{E}_s$  on the surface in terms of  $\mathbf{E}_i$  can be obtained by using Equation (28) in connection with Equation (27):

$$\mathbf{E}_s = \mathbf{E}_i - 2\mathbf{n} \times (\mathbf{E}_i \times \mathbf{n}). \quad (29)$$

A boundary condition for the magnetic vector at the scattering surface can similarly be derived:

$$\mathbf{H}_s = \mathbf{H}_i - 2(\mathbf{n} \cdot \mathbf{H}_i) \mathbf{n}. \quad (30)$$

The scattered electromagnetic field in the optical limit at an arbitrary point in space can now be derived which can be used to compute the scattering cross section of a conducting body. First a power series of the form

$$\begin{aligned} \mathbf{E} &= e^{jk\psi} \sum_{n=0}^{\infty} \frac{1}{\omega^n} \mathbf{E}_n, \\ \mathbf{H} &= e^{jk\psi} \sum_{n=0}^{\infty} \frac{1}{\omega^n} \mathbf{H}_n, \end{aligned} \quad (31)$$

can be used for the high frequency asymptotic expansion. These equations imply that a field can be made up of a sum of fields. Each of these separate waves are generated by

the primary field, or by sources, and by the interaction of these with the scattering body or parts of the scattering body [11]. Next the above equations can be substituted into Maxwell's equations giving the following expressions:

$$\begin{aligned}
 (\epsilon\mu)^{1/2} \nabla\psi \times \mathbf{E}_0 - \mu\mathbf{H}_0 &= 0, \\
 (\epsilon\mu)^{1/2} \nabla\psi \times \mathbf{H}_0 + \epsilon\mathbf{E}_0 &= 0, \\
 (\epsilon\mu)^{1/2} \nabla\psi \times \mathbf{E}_n - \mu\mathbf{H}_n &= j\nabla \times \mathbf{E}_{n-1}, \\
 (\epsilon\mu)^{1/2} \nabla\psi \times \mathbf{H}_n + \epsilon\mathbf{E}_n &= j\nabla \times \mathbf{H}_{n-1}.
 \end{aligned} \tag{32}$$

$\mathbf{E}_0$  and  $\mathbf{H}_0$  are orthogonal to  $\nabla\psi$  which can be shown by taking the dot product of  $\nabla\psi$  with the equations of zero order in the above equations. Also,  $\mathbf{E}_0$  and  $\mathbf{H}_0$  are orthogonal to each other and can be shown by taking the dot product of  $\mathbf{E}_0$  with the first relation in Equation (32). Therefore, the following orthogonality relations hold:

$$\nabla\psi \cdot \mathbf{E}_0 = \nabla\psi \cdot \mathbf{H}_0 = \mathbf{E}_0 \cdot \mathbf{H}_0 = 0 \tag{33}$$

Using the first equation of Equation (32),  $\mathbf{H}_0$  can be solved in terms of  $\nabla\psi$  and  $\mathbf{E}_0$  and substituted into the second equation of zero order to obtain,

$$\begin{aligned}
 \mathbf{H}_0 &= \left( \frac{\epsilon}{\mu} \right)^{1/2} \nabla\psi \times \mathbf{E}_0, \\
 [1 - (\nabla\psi)^2] \mathbf{E}_0 &= 0,
 \end{aligned} \tag{34}$$

where the orthogonality of  $\nabla\psi$  and  $\mathbf{E}_0$  as stated in Equation (33) was used. If  $(\nabla\psi)^2 = 1$ , the second equation in Equation (34) can be satisfied for  $\mathbf{E}_0$  that is not equal to zero.

Starting with the last two Equations in (32) one can obtain the following:

$$\nabla\psi \times \mathbf{E}_1 - \left( \frac{\mu}{\epsilon} \right)^{1/2} \mathbf{H}_1 = jc\nabla \times \mathbf{E}_0 \tag{35}$$

and

$$\nabla\psi \times \mathbf{H}_1 + \left(\frac{\varepsilon}{\mu}\right)^{1/2} \mathbf{E}_1 = jc \nabla \times \mathbf{H}_0. \quad (36)$$

Next, solve the first equation above for  $\mathbf{H}_1$  in terms of  $\mathbf{E}_1$  and  $\mathbf{E}_0$  to get

$$\mathbf{H}_1 = \left(\frac{\mu}{\varepsilon}\right)^{-1/2} (\nabla\psi \times \mathbf{E}_1) - \frac{j}{\mu} \nabla \times \mathbf{E}_0, \quad (37)$$

and solve the first equation of Equation (32) for  $\mathbf{H}_0$  in terms of  $\mathbf{E}_0$  to get

$$\mathbf{H}_0 = \left(\frac{\varepsilon}{\mu}\right)^{1/2} \nabla\psi \times \mathbf{E}_0. \quad (38)$$

Now using Equation (36), substitute Equations (37) and (38) and use the vector identities, the result is

$$\alpha \nabla\psi = \nabla\psi \times (\nabla \times \mathbf{E}_0) - (\nabla^2\psi) \mathbf{E}_0 - (\nabla\psi \cdot \nabla) \mathbf{E}_0 + (\mathbf{E}_0 \cdot \nabla) \nabla\psi \quad (39)$$

where  $\alpha$  is a scalar quantity [11]. Equation (39) can be transformed into a simple ordinary differential equation for  $\mathbf{E}_0$ . Applying the orthogonality relationship of Equation (33) and the fact that the curl of a gradient is zero to the vector identity that gives the expression of the gradient of the dot product of two vectors  $\nabla\psi$  and  $\mathbf{E}_0$  we find

$$\nabla\psi \times (\nabla \times \mathbf{E}_0) = -(\nabla\psi \cdot \nabla) \mathbf{E}_0 - (\mathbf{E}_0 \cdot \nabla) \nabla\psi. \quad (40)$$

Substituting Equation (40) into Equation (39)

$$\alpha \nabla\psi = -2(\nabla\psi \cdot \nabla) \mathbf{E}_0 - (\nabla^2\psi) \mathbf{E}_0. \quad (41)$$

Looking at the first term on the righthand side of Equation (41), it can be interpreted as the ordinary derivative with respect to arc length  $s$  along the ray because of the relation

$$\nabla\psi \cdot \nabla = \frac{d}{ds} \quad (42)$$

therefore,

$$\alpha \nabla\psi = -2 \frac{d\mathbf{E}_0}{ds} - (\nabla^2\psi) \mathbf{E}_0. \quad (43)$$

By taking the dot product of both sides with the quantity  $\nabla\psi$ , and using Equation (33)

and  $(\nabla\psi^2)=1$ , the quantity  $\alpha$  can be determined

$$\alpha = -2 \nabla\psi \cdot \frac{d\mathbf{E}_0}{ds}, \quad (44)$$

and since

$$0 = \frac{d}{ds} (\nabla\psi \cdot \mathbf{E}_0) = \mathbf{E}_0 \cdot \frac{d}{ds} (\nabla\psi) + \nabla\psi \cdot \frac{d\mathbf{E}_0}{ds} \quad (45)$$

the quantity  $\alpha$  becomes

$$\alpha = 2 \mathbf{E}_0 \cdot \frac{d}{ds} \nabla\psi. \quad (46)$$

Because of Equation (42), the equation for  $\alpha$  can be written as

$$\alpha = 2 \mathbf{E}_0 \cdot [\nabla\psi \cdot \nabla(\nabla\psi)] \quad (47)$$

Using the fact that  $(\nabla\psi^2)=1$ , one can find that

$$0 = \nabla(1) = \nabla[(\nabla\psi)^2] = 2 \nabla\psi \cdot \nabla(\nabla\psi). \quad (48)$$

From the two equations above, one can see that  $\alpha$  is equal to zero and therefore

Equation (43) becomes

$$2 \frac{d\mathbf{E}_0}{ds} + (\nabla^2\psi) \mathbf{E}_0 = 0. \quad (49)$$

Equation (49) is an ordinary differential equation along a ray for the electric field strength  $\mathbf{E}_0$  corresponding to the lowest-order (geometrical-optics) approximation for the limit of small wavelengths [11]. The  $\mathbf{E}_0$  field vector can be expressed as

$$\mathbf{E}_0 = \mathbf{A} \exp\left(-\frac{1}{2} \int_{s_0}^s \nabla^2 \psi ds\right) \quad (50)$$

where  $\mathbf{A}$  is a constant vector that is the value of  $\mathbf{E}_0$  at the initial point determined by  $s=s_0$  on the ray. If one considers a length of ray running from the point  $s_0$  to an arbitrary point  $s_1$  and surrounded by an infinitesimal tube of parallel rays with wavefront area elements  $d\sigma_0$  and  $d\sigma_1$  capping the tube at the points  $s_0$  and the point  $s_1$ ,  $\mathbf{E}_0$  can be written in a simpler form. Next consider the volume integral over this infinitesimal tube of rays

$$I = \iiint_V \nabla^2 \psi dv. \quad (51)$$

Using Gauss's theorem,

$$I = \int_S (\nabla \psi) \cdot \mathbf{n} d\sigma, \quad (52)$$

where  $S$  is the surface surrounding the infinitesimal tube of rays and  $\mathbf{n}$  is the outward-pointing unit normal to the surface. The integrand of Equation (52) will be zero except on that part of  $S$  that consists of the two wavefront caps since  $\nabla \psi$  has the direction of a ray. The volume integral becomes

$$I = \int_{s_1} \int \frac{d\psi}{ds} d\sigma_1 - \int_{s_0} \int \frac{d\psi}{ds} d\sigma_0 = \int_{s_0} \int \left( \frac{d\sigma_1}{d\sigma_0} - 1 \right) d\sigma_0 \quad (53)$$

where  $s_1$  is the wavefront cap at the point  $s_1$  and  $s_0$  is the wavefront cap at the point  $s_0$ .

The quantity  $d\sigma_1/d\sigma_0$  is the Jacobian of the point-to-point transformation from the surface

$S_0$  to the surface  $S_1$  given by the rays passing through both surfaces [11]. The equation for  $I$  can be rewritten as

$$I = \iint_{S_0} \int_{S_0}^{S_1} \frac{d}{ds} \left( \frac{d\sigma_1}{d\sigma_0} \right) ds d\sigma_0 = \int_{S_0}^{S_1} \iint_{S_0} \frac{d}{ds} \left( \frac{d\sigma_1}{d\sigma_0} \right) d\sigma_0 ds. \quad (54)$$

The surface integration can be taken over the surface  $S_1$  so that

$$I = \int_{S_0}^{S_1} \int_{S_1} \int \frac{d\sigma_0}{d\sigma_1} \frac{d}{ds} \left( \frac{d\sigma_1}{d\sigma_0} \right) d\sigma ds = \iint_V \int \frac{d}{ds} \log \left( \frac{d\sigma_1}{d\sigma_0} \right) dv. \quad (55)$$

Now comparing Equation (55) with Equation (51), one can see that

$$\nabla^2 \psi = \frac{d}{ds} \log \left( \frac{d\sigma_1}{d\sigma_0} \right). \quad (56)$$

Substituting Equation (56) into Equation (50)

$$\mathbf{E}_0 = \mathbf{A} \left( \frac{d\sigma_0}{d\sigma_1} \right)^{1/2} \quad (57)$$

where  $d\sigma_0/d\sigma_1$  is the reciprocal of the Jacobian of the point-to-point transformation. The expression for the magnetic field vector can be obtained from Equation (33) and Equation (57)

$$\mathbf{H}_0 = \left( \frac{\epsilon}{\mu} \right)^{1/2} \nabla \psi \times \mathbf{A} \left( \frac{d\sigma_0}{d\sigma} \right)^{1/2}. \quad (58)$$

It can be shown that the energy on a wavefront is proportional to the quantity  $d\sigma_0/d\sigma_1$  which is important for computing the geometrical-optics scattering cross section of a body [11].

Now, if the boundary conditions in Equations (29) and (30) are applied to the zero-order term in Equation (31) for the electromagnetic field vectors, along with Equation (57), the scattered electromagnetic field in the optical limit at an arbitrary point in space will be given by

$$\mathbf{E}_s = [\mathbf{E}_i - 2\mathbf{n} \times (\mathbf{E}_i \times \mathbf{n})] \left( \frac{d\sigma_0}{d\sigma} \right)^{1/2} e^{jks} \quad (59)$$

and

$$\mathbf{H}_s = [\mathbf{H}_i - 2(\mathbf{n} \cdot \mathbf{H}_i)\mathbf{n}] \left( \frac{d\sigma_0}{d\sigma} \right)^{1/2} e^{jks} \quad (60)$$

where the quantity  $s$  is the arc length measured from the reflection point on the scattering body surface to the point of observation along the reflected ray through the point [11]. The reflection on the surface of the scattering body is the surface element  $d\sigma_0$  and the reflected wavefront passing through the point of observation is the surface element  $d\sigma$ .

#### 4. Description of Waveforms

The impulse waveform that was used is a sinc waveform in the time domain, which becomes a rectangular waveform in the frequency domain. The rectangular waveform started at 600 MHz and had an upper frequency of 1.1 GHz with a constant amplitude. This yields a bandwidth ratio (upper frequency divided by the lower frequency) of 1.83. The impulse waveform in the time domain is shown in Figure 1. Next, a fast Fourier transform (FFT) was performed to obtain the impulse waveform in the frequency domain as shown in Figure 2.

The frequency waveform in Figure 2 shows an overshoot at the discontinuities (sharp changes), which is known as Gibbs phenomenon. In the neighborhood of points of discontinuity in  $f(t)$ , the Fourier series representation fails to converge even though the mean-square error in the representation approaches zero. The overshoot peak moves closer to the point of discontinuity as more terms in the series are added, but the overshoot still exists.

For this analysis, the second waveform, as shown in Figure 3, was a pulsed continuous wave (CW) signal centered at 1 GHz and consisted of a pulse train that was on for 25 Hz and off for 250 Hz. An FFT was also performed to obtain the frequency domain waveform as shown in Figure 4. An unmodulated CW signal was not used because it does not provide any range information. To overcome this lack of range information, the CW signal is modulated to provide a timing mark. The timing mark permits the time of transmission and the time of return to be identified. The accuracy of the measurement of the transit time can be improved with a sharper or more distinct timing mark. When the timing mark becomes distinct, the transmitted spectrum becomes

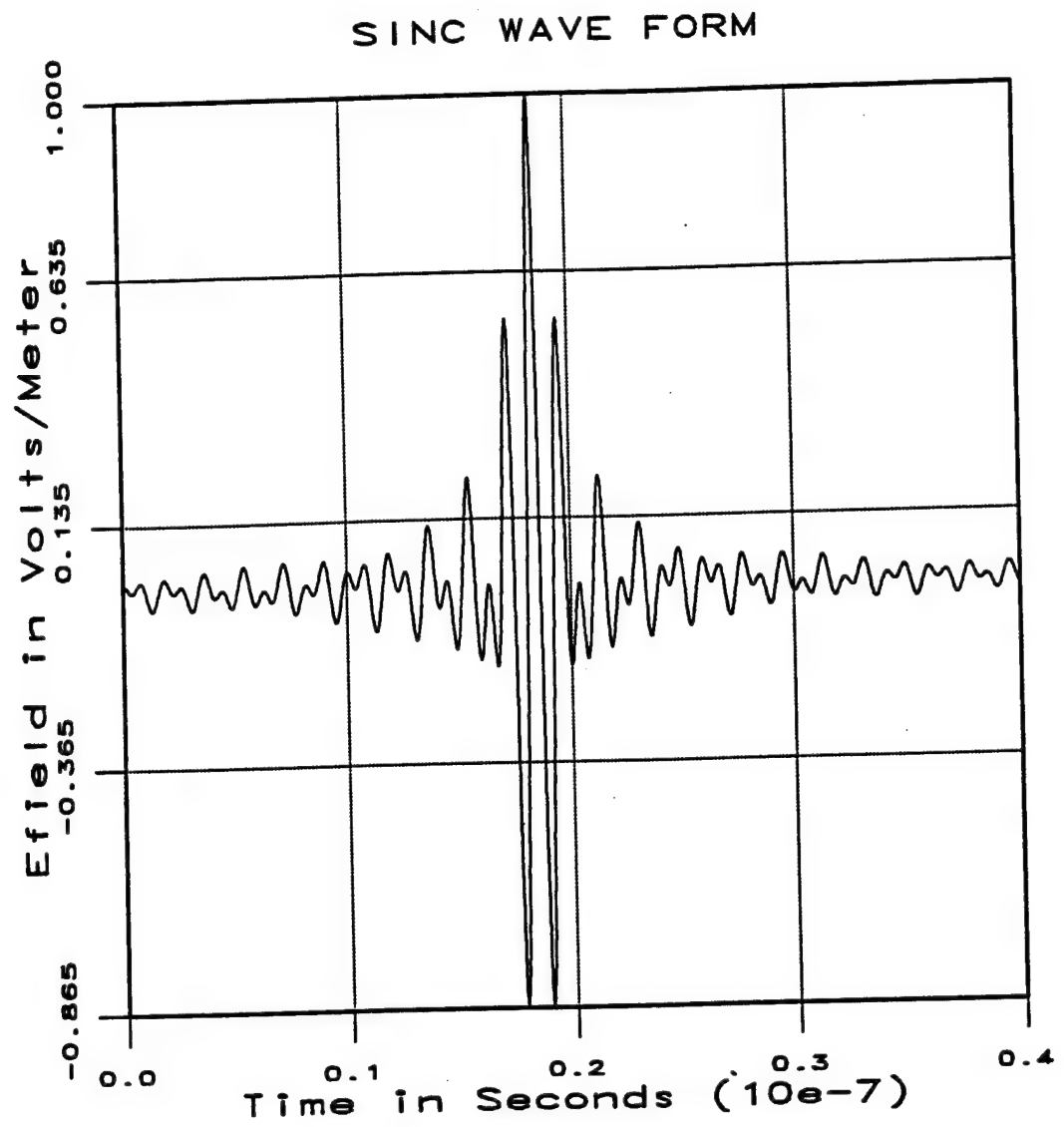


Figure 1. Graph of Impulse Time Domain Waveform.

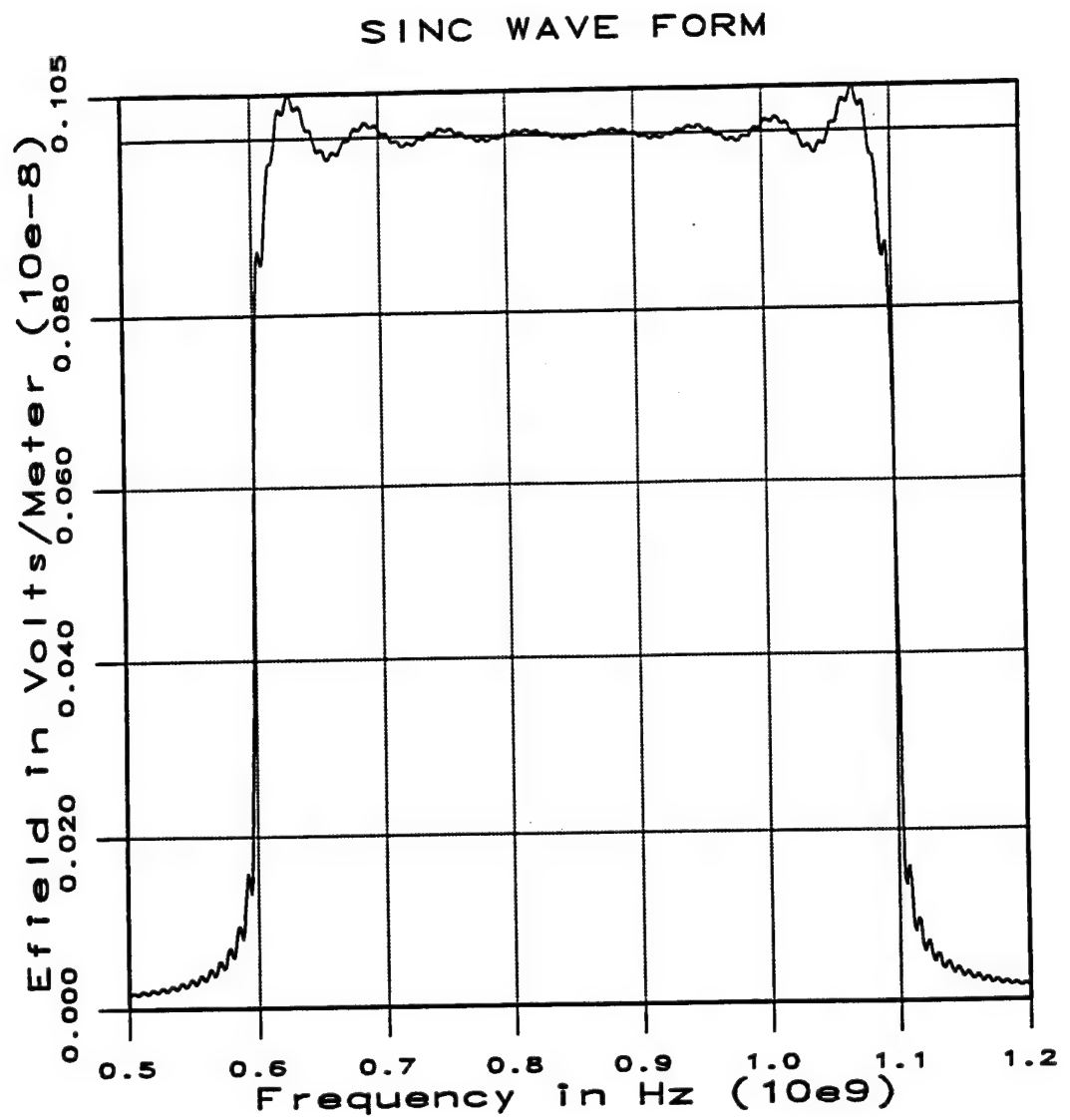


Figure 2. Graph of Impulse Frequency Domain Waveform.

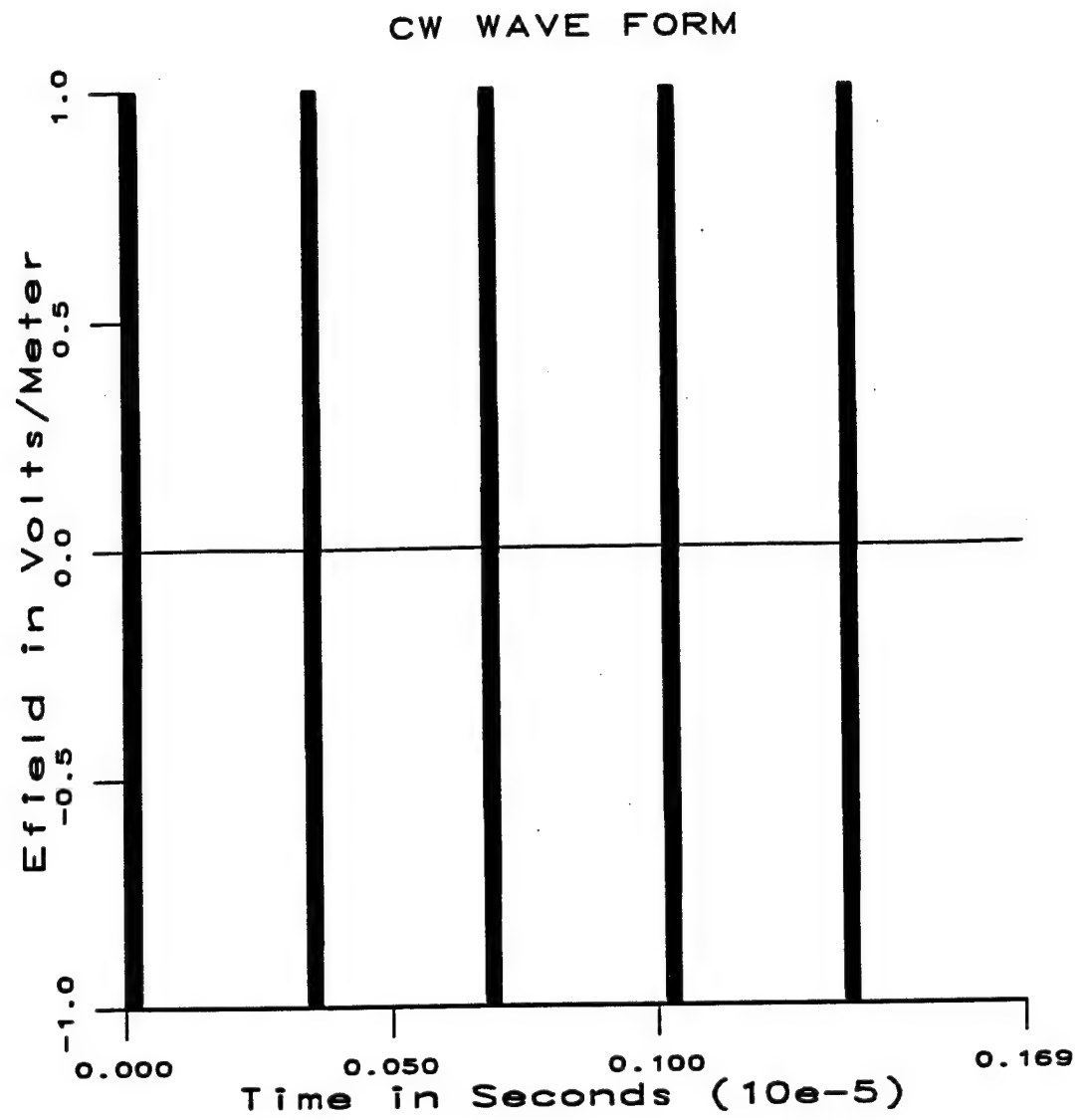


Figure 3. Graph of Pulsed CW Time Domain Waveform.

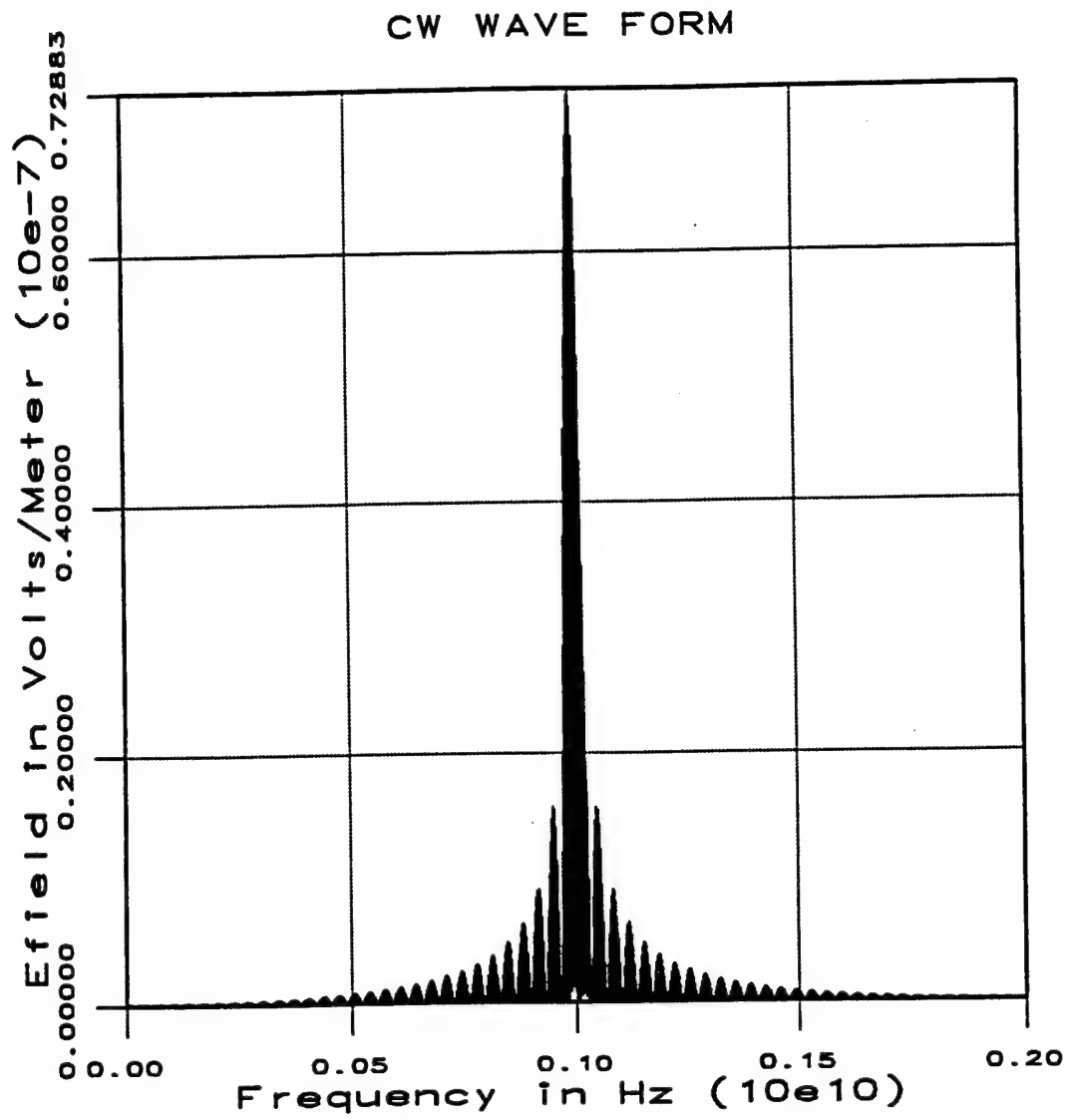


Figure 4. Graph of Pulsed CW Frequency Domain Waveform.

broadener. Therefore, a finite spectrum must be transmitted if transit time or range is to be measured [7]. Three techniques are normally used in modulating CW radars to broaden the spectrum, they are frequency, phase, and pulse coding.

For frequency modulated CW radars the frequency of the transmitted signal is varied, and the range is measured based on the frequency difference between the instantaneous transmitted and received signals. The transit time is proportional to the difference in frequency between the echo signal and the transmitter signal. The greater the transmitter frequency deviation in a given time interval, the more accurate the measurement of the transit time, and the greater will be the transmitted spectrum [7]. Typically, frequency modulations of the CW signal are triangular, saw-tooth, and sinusoidal.

For phase coding the CW signal, either a binary or a polyphase code can be used. For binary coding, the signal is divided into segments, with a +1 represented by  $0^\circ$  phase implying in-phase, and a 0 represented by a  $180^\circ$  phase shift implying out-of-phase, with respect to the carrier. The return signal can then be correlated with a delayed version of the transmitted code to produce range discrimination approximately equal to half the bit length [6].

The most common coding technique is the amplitude modulation employed in the pulse radar which will also be used in this analysis. In this technique, the range to the target is determined by measuring the time taken by the pulse to travel to the target and return. The range  $R$  is

$$R = c\Delta t / 2, \quad (61)$$

where  $c$  is the velocity of light and  $\Delta t$  is the two-way transit time.

For a pulse radar, a continuous train of pulses at some pulse repetition frequency (PRF) is transmitted. It is important that a sufficient length of time must elapse to allow any echo signals to return and be detected before the next pulse may be transmitted. Therefore the rate at which the pulses may be transmitted is determined by the longest range at which targets are expected [7]. Ambiguities in measuring range might result if the PRF were too high by having echo signals from some targets arriving before the transmission of the next pulse. These range-ambiguous targets will appear to be at a much shorter range than the actual, and could be misleading. The maximum unambiguous range which a radar can measure is given by

$$R_{\text{unamb}} = \frac{c}{2f_r} \quad (62)$$

where  $f_r$  is the PRF in Hz [6]. Pulse radars have many advantages, which enable returns over a specific range interval to be selected and recorded, and unwanted signals from other range intervals to be discriminated against, based upon their time of arrival.

For this analysis a Gaussian waveform (see Figure 5) is used as a baseline waveform. The frequency domain waveform, as shown in Figure 6, implies a very wide bandwidth.

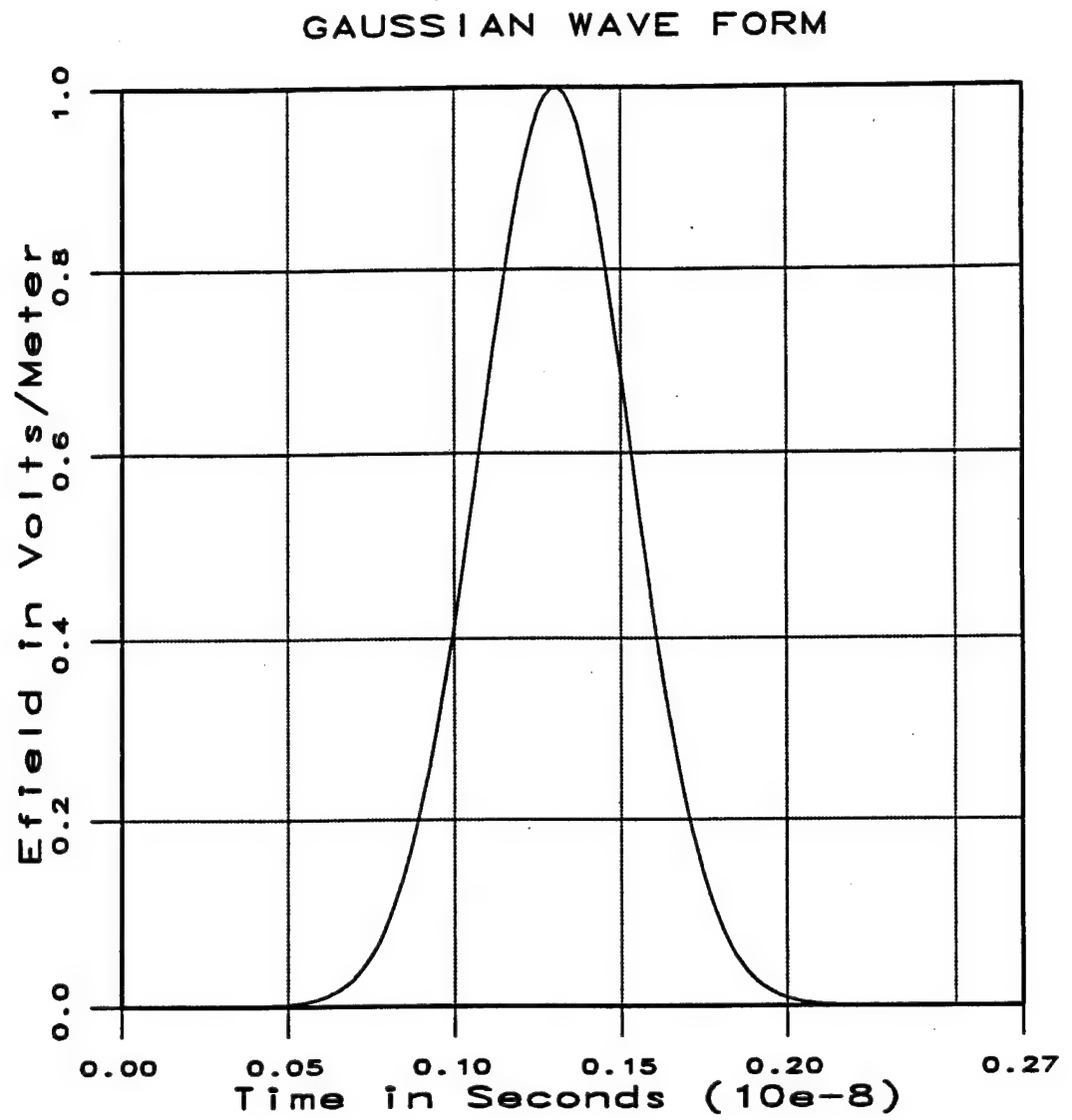


Figure 5. Graph of Gaussian Time Domain Waveform.

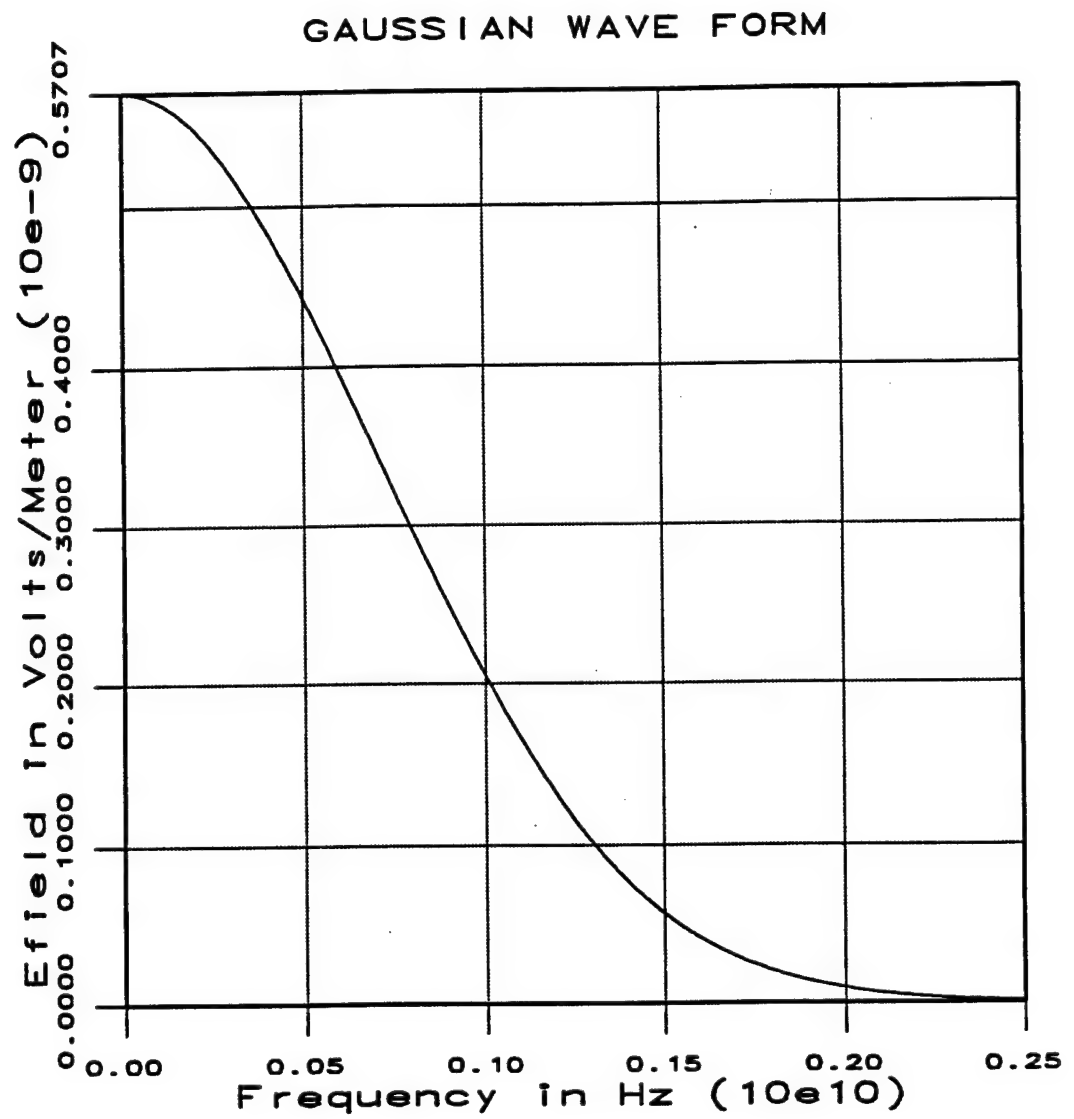


Figure 6. Graph of Gaussian Frequency Domain Waveform.

## 5. Finite Difference Time Domain Analysis

To investigate the effects of change of target geometry and incident pulse shape, one can use Finite Difference Time Domain (FDTD) techniques. The scattered field FDTD formulation starts by examining Maxwell's equations in a linear medium:

$$\begin{aligned}\nabla \times \mathbf{E} &= -\partial \mathbf{B} / \partial t \\ \nabla \times \mathbf{H} &= \partial \mathbf{D} / \partial t + \mathbf{J} \\ \nabla \cdot \mathbf{D} &= \rho \\ \nabla \cdot \mathbf{B} &= 0\end{aligned}\tag{63}$$

where

$$\begin{aligned}\mathbf{D} &= \epsilon \mathbf{E} \\ \mathbf{B} &= \mu \mathbf{H}\end{aligned}\tag{64}$$

The FDTD formulation only requires the Maxwell's curl equations because the divergence equations are redundant in that they are contained within the curl equations and the initial conditions. To show this, one takes the divergence of the curl equations to obtain

$$\nabla \cdot (\nabla \times \mathbf{E}) = -\frac{\partial}{\partial t} (\nabla \cdot \mathbf{B})\tag{65}$$

and since divergence of a curl is zero, then

$$-\partial (\nabla \cdot \mathbf{B}) / \partial t = 0\tag{66}$$

implying that

$$\nabla \cdot \mathbf{B} = \text{constant}\tag{67}$$

and

$$\nabla \cdot (\nabla \times \mathbf{H}) = \frac{\partial}{\partial t} (\nabla \cdot \mathbf{D}) + \nabla \cdot \mathbf{J}\tag{68}$$

where again divergence of a curl is zero, so

$$\frac{\partial}{\partial t}(\nabla \cdot \mathbf{D}) + \nabla \cdot \mathbf{J} = 0 \quad (69)$$

and the continuity equation is

$$\nabla \cdot \mathbf{J} + \partial \rho / \partial t = 0 \quad (70)$$

therefore

$$\partial (\nabla \cdot \mathbf{D}) / \partial t - \partial \rho / \partial t = 0 \quad (71)$$

$$\frac{\partial}{\partial t}[(\nabla \cdot \mathbf{D}) - \rho] = 0 \quad (72)$$

implying that

$$\nabla \cdot \mathbf{D} - \rho = \text{constant} . \quad (73)$$

The above equations used the vector identity  $\nabla \cdot \nabla \times \mathbf{A} = 0$ . The field and sources are set to zero at the initial time implying that  $\nabla \cdot \mathbf{B}$  and  $(\nabla \cdot \mathbf{D} - \rho)$  must be zero for all times [12].

Therefore, the curl equations can then be rewritten in the following form:

$$\begin{aligned} \mathbf{H} / \partial t &= -\frac{1}{\mu}(\nabla \times \mathbf{E}) - \frac{\sigma^*}{\mu} \mathbf{H} \\ \mathbf{E} / \partial t &= \frac{-\sigma}{\epsilon} \mathbf{E} + \frac{1}{\epsilon}(\nabla \times \mathbf{H}) \end{aligned} \quad (74)$$

where  $\mathbf{J} = \sigma \mathbf{E}$  to allow for lossy dielectric material and also included the possibility of magnetic loss by adding a magnetic conductivity term  $\sigma^*$ . Any linear isotropic material property can be specified because all four constitutive parameters  $\epsilon$ ,  $\mu$ ,  $\sigma$ , and  $\sigma^*$  are present.

The  $\mathbf{E}$  and  $\mathbf{H}$  fields can be expressed as

$$\mathbf{E} = \mathbf{E}^{\text{total}} \equiv \mathbf{E}^{\text{incident}} + \mathbf{E}^{\text{scattered}} \quad (75)$$

and

$$\mathbf{H} = \mathbf{H}^{\text{total}} \equiv \mathbf{H}^{\text{incident}} + \mathbf{H}^{\text{scattered}} \quad (76)$$

The incident field components can be specified analytically throughout the problem space and the scattered fields are found computationally. Only the scattered fields are required to be absorbed at the problem space outer boundaries. The scattered wave arises on and within the scattering object in response to the incident field so as to satisfy the appropriate boundary conditions, which are the Maxwell equations themselves, on or within the scattering object. For a perfect conductor, it is required that  $\mathbf{E}^{\text{scattered}} = -\mathbf{E}^{\text{incident}}$  in the scatterer. For a nonperfect conductor, the scattered fields depend on the constitutive parameters of the material. When in this medium, the scattered fields are subject to the Maxwell equations, while outside this media they satisfy the free space Maxwell equations. The field that is defined as being present in the absence of the scatterer is the incident field because it always propagates in free space even when passing through the scatterer material [12].

One criteria that is required is that the incident and scattered fields must satisfy the Maxwell equations independently. The total field propagates in free space outside the scatterer and in the media of the scatterer when it is propagating within the scatterer. While the incident field travels through free space throughout the problem space. The total fields satisfy the following equations in the media of the scatterer:

$$\nabla \times \mathbf{E}^{\text{total}} = -\mu \partial \mathbf{H}^{\text{total}} / \partial t - \sigma \cdot \mathbf{H}^{\text{total}} \quad (77)$$

and

$$\nabla \times \mathbf{H}^{\text{total}} = -\epsilon \partial \mathbf{E}^{\text{total}} / \partial t + \sigma \mathbf{E}^{\text{total}} . \quad (78)$$

The incident fields traversing the media satisfy free space conditions

$$\nabla \times \mathbf{E}^{\text{inc}} = -\mu_0 \partial \mathbf{H}^{\text{inc}} / \partial t \quad (79)$$

and

$$\nabla \times \mathbf{H}^{\text{inc}} = \epsilon_0 \partial \mathbf{E}^{\text{inc}} / \partial t. \quad (80)$$

Rewriting Equations (77) and (78) one finds the following:

$$\nabla \times (\mathbf{E}^{\text{inc}} + \mathbf{E}^{\text{scat}}) = -\mu \partial (\mathbf{H}^{\text{inc}} + \mathbf{H}^{\text{scat}}) / \partial t - \sigma^* (\mathbf{H}^{\text{inc}} + \mathbf{H}^{\text{scat}}) \quad (81)$$

and

$$\nabla \times (\mathbf{H}^{\text{inc}} + \mathbf{H}^{\text{scat}}) = \epsilon \partial (\mathbf{E}^{\text{inc}} + \mathbf{E}^{\text{scat}}) / \partial t + \sigma (\mathbf{E}^{\text{inc}} + \mathbf{E}^{\text{scat}}). \quad (82)$$

To obtain the equations governing the scattered fields in the media, the incident fields are subtracted from Equations (81) and (82).

$$\nabla \times \mathbf{E}^{\text{scat}} = -\mu \partial \mathbf{H}^{\text{scat}} / \partial t - \sigma^* \mathbf{H}^{\text{scat}} - [(\mu - \mu_0) \partial \mathbf{H}^{\text{inc}} / \partial t - \sigma^* \mathbf{H}^{\text{inc}}] \quad (83)$$

and

$$\nabla \times \mathbf{H}^{\text{scat}} = \epsilon \partial \mathbf{E}^{\text{scat}} / \partial t + \sigma \mathbf{E}^{\text{scat}} + [(\epsilon - \epsilon_0) \partial \mathbf{E}^{\text{inc}} / \partial t + \sigma \mathbf{E}^{\text{inc}}] . \quad (84)$$

The total fields satisfy, outside the scatterer in free space, the following relationship:

$$\nabla \times \mathbf{E}^{\text{total}} = -\mu_0 \partial \mathbf{H}^{\text{total}} / \partial t \quad (85)$$

and

$$\nabla \times \mathbf{H}^{\text{total}} = \epsilon_0 \partial \mathbf{E}^{\text{total}} / \partial t \quad (86)$$

which can also be written as

$$\nabla \times (\mathbf{E}^{\text{inc}} + \mathbf{E}^{\text{scat}}) = -\mu_0 \partial (\mathbf{H}^{\text{inc}} + \mathbf{H}^{\text{scat}}) / \partial t \quad (87)$$

and

$$\nabla \times (\mathbf{H}^{\text{inc}} + \mathbf{H}^{\text{scat}}) = \epsilon_0 \partial (\mathbf{E}^{\text{inc}} + \mathbf{E}^{\text{scat}}) / \partial t. \quad (88)$$

To obtain the equations governing the scattered fields in free space, subtract the incident fields from the above equations to find the following:

$$\nabla \times \mathbf{E}^{\text{scat}} = -\mu_0 \partial \mathbf{H}^{\text{scat}} / \partial t \quad (89)$$

and

$$\nabla \times \mathbf{H}^{\text{scat}} = \epsilon_0 \partial \mathbf{E}^{\text{scat}} / \partial t. \quad (90)$$

Equations (83) and (84) can now be rearranged so that the time derivative of the field is expressed as a function of the remaining terms for ease in generating the appropriate difference equations

$$\frac{\partial \mathbf{H}^{\text{scat}}}{\partial t} = \frac{-\sigma^*}{\mu} \mathbf{H}^{\text{scat}} - \frac{\sigma^*}{\mu} \mathbf{H}^{\text{inc}} - \frac{(\mu - \mu_0)}{\mu} \frac{\partial \mathbf{H}^{\text{inc}}}{\partial t} - \frac{1}{\mu} (\nabla \times \mathbf{E}^{\text{scat}}) \quad (91)$$

and

$$\frac{\partial \mathbf{E}^{\text{scat}}}{\partial t} = \frac{-\sigma}{\epsilon} \mathbf{E}^{\text{scat}} - \frac{\sigma}{\epsilon} \mathbf{E}^{\text{inc}} - \frac{(\epsilon - \epsilon_0)}{\epsilon} \frac{\partial \mathbf{E}^{\text{inc}}}{\partial t} - \frac{1}{\epsilon} (\nabla \times \mathbf{H}^{\text{scat}}). \quad (92)$$

The perfect conductor will first be analyzed for the finite difference time domain formulation. First look at the scattered fields outside the scatterer which satisfy the free space conditions where  $\sigma^* = \sigma = 0$ ,  $\mu = \mu_0$ , and  $\epsilon = \epsilon_0$ . Equations (91) and (92) therefore reduce to

$$\frac{\partial \mathbf{H}^{\text{scat}}}{\partial t} = -\frac{1}{\mu_0} (\nabla \times \mathbf{E}^{\text{scat}}) \quad (93)$$

and

$$\frac{\partial \mathbf{E}^{\text{scat}}}{\partial t} = \frac{1}{\epsilon_0} (\nabla \times \mathbf{H}^{\text{scat}}). \quad (94)$$

Next, in the perfect conductor the scattered E-field can be written as

$$\frac{\epsilon}{\sigma} \frac{\partial \mathbf{E}^{\text{scat}}}{\partial t} = -\mathbf{E}^{\text{scat}} - \mathbf{E}^{\text{inc}} - \frac{(\epsilon - \epsilon_0)}{\sigma} \frac{\partial \mathbf{E}^{\text{inc}}}{\partial t} + \frac{1}{\sigma} (\nabla \times \mathbf{H}^{\text{scat}}). \quad (95)$$

For a perfect conductor  $\sigma = \infty$ . Substituting  $\sigma = \infty$  into Equation (95), one gets

$$\mathbf{E}^{\text{scat}} = -\mathbf{E}^{\text{inc}}. \quad (96)$$

Equation (96) need only be applied at the surface of the perfect conductor because interior portions of the perfect conductor, if present, are completely isolated from the rest of the problem space [12]. The free space scattered field equations can now be differenced. This is done by replacing derivatives with differences

$$\frac{\partial f}{\partial t} \equiv \lim_{\Delta t \rightarrow 0} \frac{f(x, t_2) - f(x, t_1)}{\Delta t} \approx \frac{f(x, t_2) - f(x, t_1)}{\Delta t} \quad (97)$$

and

$$\frac{\partial f}{\partial x} \equiv \lim_{\Delta x \rightarrow 0} \frac{f(x_2, t) - f(x_1, t)}{\Delta x} \approx \frac{f(x_2, t) - f(x_1, t)}{\Delta x} \quad (98)$$

where the approximations  $\Delta t = t_2 - t_1$  and  $\Delta x = x_2 - x_1$  and both are finite rather than infinitesimal. An explicit central difference scheme will be used that only retains first order terms and the  $\mathbf{E}$  and  $\mathbf{H}$  fields are interweaved spatially and temporally.

The vector Maxwell curl equations governing the scattered fields will be decomposed into their component scalar parts, obtaining

$$\frac{\partial \mathbf{E}_x^{\text{scat}}}{\partial t} = \frac{1}{\epsilon_0} \left( \frac{\partial \mathbf{H}_z^{\text{scat}}}{\partial y} - \frac{\partial \mathbf{H}_y^{\text{scat}}}{\partial z} \right) \quad (99)$$

$$\frac{\partial \mathbf{E}_y^{\text{scat}}}{\partial t} = \frac{1}{\epsilon_0} \left( \frac{\partial \mathbf{H}_x^{\text{scat}}}{\partial z} - \frac{\partial \mathbf{H}_z^{\text{scat}}}{\partial x} \right) \quad (100)$$

$$\frac{\partial \mathbf{E}_z^{\text{scat}}}{\partial t} = \frac{1}{\epsilon_0} \left( \frac{\partial \mathbf{H}_y^{\text{scat}}}{\partial x} - \frac{\partial \mathbf{H}_x^{\text{scat}}}{\partial y} \right) \quad (101)$$

$$\frac{\partial \mathbf{H}_x^{\text{scat}}}{\partial t} = \frac{1}{\mu_0} \left( \frac{\partial \mathbf{E}_y^{\text{scat}}}{\partial z} - \frac{\partial \mathbf{E}_z^{\text{scat}}}{\partial y} \right) \quad (102)$$

$$\frac{\partial \mathbf{H}_y^{\text{scat}}}{\partial t} = \frac{1}{\mu_0} \left( \frac{\partial \mathbf{E}_z^{\text{scat}}}{\partial x} - \frac{\partial \mathbf{E}_x^{\text{scat}}}{\partial z} \right) \quad (103)$$

$$\frac{\partial \mathbf{H}_z^{\text{scat}}}{\partial t} = \frac{1}{\mu_0} \left( \frac{\partial \mathbf{E}_x^{\text{scat}}}{\partial y} - \frac{\partial \mathbf{E}_y^{\text{scat}}}{\partial x} \right) \quad (104)$$

Replacing with differences results in

$$\frac{\mathbf{E}_x^{\text{scat},n} - \mathbf{E}_x^{\text{scat},n-1}}{\Delta t} = \frac{1}{\epsilon_0} \left[ \frac{\Delta \mathbf{H}_z^{\text{scat},n-1/2}}{\Delta y} - \frac{\Delta \mathbf{H}_y^{\text{scat},n-1/2}}{\Delta z} \right] \quad (105)$$

$$\frac{\mathbf{E}_y^{\text{scat},n} - \mathbf{E}_y^{\text{scat},n-1}}{\Delta t} = \frac{1}{\epsilon_0} \left[ \frac{\Delta \mathbf{H}_x^{\text{scat},n-1/2}}{\Delta z} - \frac{\Delta \mathbf{H}_z^{\text{scat},n-1/2}}{\Delta x} \right] \quad (106)$$

$$\frac{\mathbf{E}_z^{\text{scat},n} - \mathbf{E}_z^{\text{scat},n-1}}{\Delta t} = \frac{1}{\epsilon_0} \left[ \frac{\Delta \mathbf{H}_y^{\text{scat},n-1/2}}{\Delta x} - \frac{\Delta \mathbf{H}_x^{\text{scat},n-1/2}}{\Delta y} \right] \quad (107)$$

$$\frac{\mathbf{H}_x^{\text{scat},n+1/2} - \mathbf{H}_x^{\text{scat},n-1/2}}{\Delta t} = \frac{1}{\mu_0} \left[ \frac{\Delta \mathbf{E}_y^{\text{scat},n}}{\Delta z} - \frac{\Delta \mathbf{E}_z^{\text{scat},n}}{\Delta y} \right] \quad (108)$$

$$\frac{\mathbf{H}_y^{\text{scat},n+1/2} - \mathbf{H}_y^{\text{scat},n-1/2}}{\Delta t} = \frac{1}{\mu_0} \left[ \frac{\Delta \mathbf{E}_z^{\text{scat},n}}{\Delta x} - \frac{\Delta \mathbf{E}_x^{\text{scat},n}}{\Delta z} \right] \quad (109)$$

$$\frac{\mathbf{H}_z^{\text{scat},n+1/2} - \mathbf{H}_z^{\text{scat},n-1/2}}{\Delta t} = \frac{1}{\mu_0} \left[ \frac{\Delta \mathbf{E}_x^{\text{scat},n}}{\Delta y} - \frac{\Delta \mathbf{E}_y^{\text{scat},n}}{\Delta x} \right] \quad (110)$$

Now the above equations can be put into the form used in a perfectly conducting version of a FDTD code. The first step is to quantify space by letting  $x = I\Delta x$ ,  $y = J\Delta y$ ,

and  $z = K\Delta z$ , and quantify time by letting  $t = n\Delta t$ . Next, uniform cells are defined in the problem space and can be located by the I, J, and K indices. As shown in Figure 5, the “Yee cell” has the field components located at the offsets. In Yee notation  $E_z^n(I,J,K)$  represents the z component of the electric field at time  $t = n\Delta t$  and at spatial location  $x = I\Delta x$ ,  $y = J\Delta y$ , and  $z = (K+1/2)\Delta z$  [13].

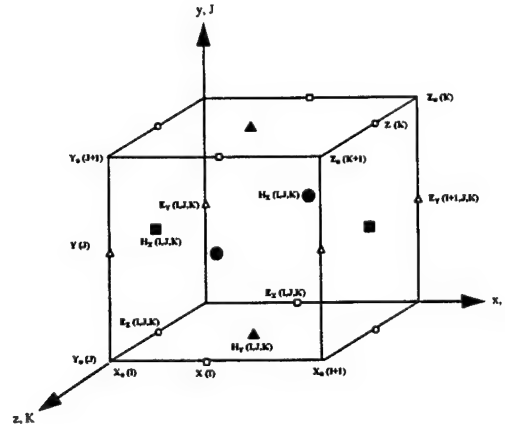


Figure 7. Location of the Six-field Evaluation Points in a Typical Yee Cell.

## 6. Description of the Targets

In this analysis, three objects are identified using the impulse waveform and a pulsed CW waveform, and a comparison between the two waveforms for the three objects is made. The first object is a simple perfect conductor sphere shaped object, as shown in Figure 8. The second object is a perfect conducting square plate, as shown in Figure 9, and the third object is an elliptical shape with a cylinder rod through the center, as shown in Figure 10. The objects are constructed in Yee cells. The first important determination is the cell size. The cell size must be small enough to permit accurate results at the highest frequency of interest, but large enough to keep the computational time manageable. The fundamental constraint is that the side of each cell should be  $1/10\lambda$  or less at the highest frequency (shortest wavelength) of interest [12]. For this analysis, the highest frequency is 1.1 GHz, which gives a cell size of 0.027m. However, to resolve the shape of the sphere the cell size needed to be reduced to 0.014m. The radius of the sphere was 19.5 cells with a 50 x 50 x 50 mesh. The square plate is 30 x 30 cells with a 60 x 60 x 60 mesh.

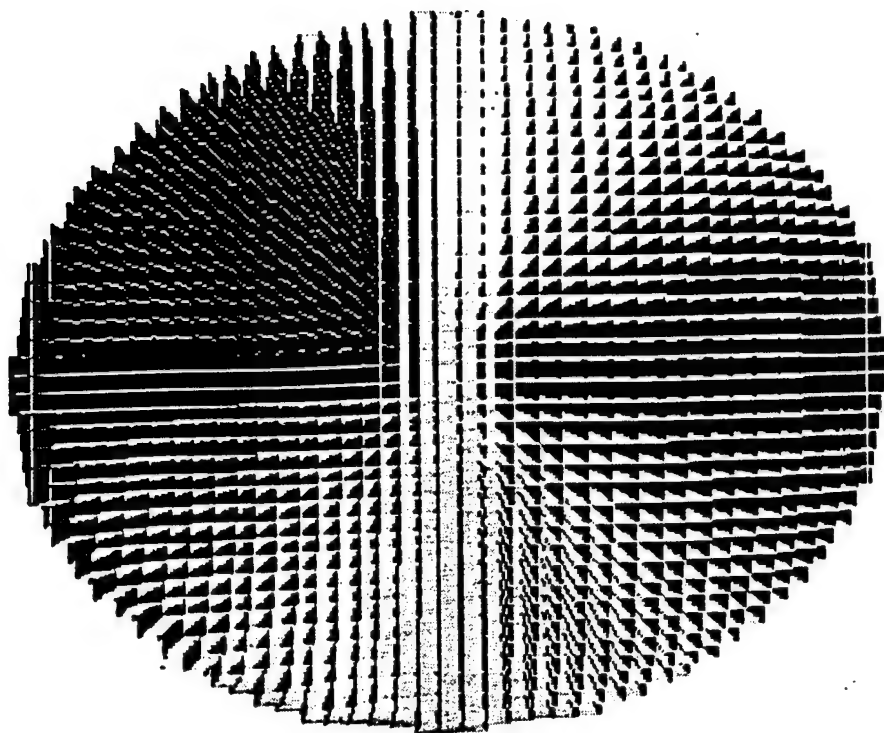


Figure 8. Perfect Conducting Sphere. Radius = 19.5 cells. Cell Size = 0.014m.  
50 x 50 x 50 Cell Mesh.

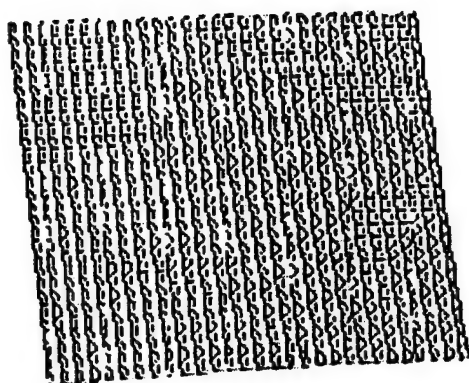


Figure 9. Perfect Conducting Square Plate. 30 x 30 Cells. 60 x 60 x 60 Cell Mesh.

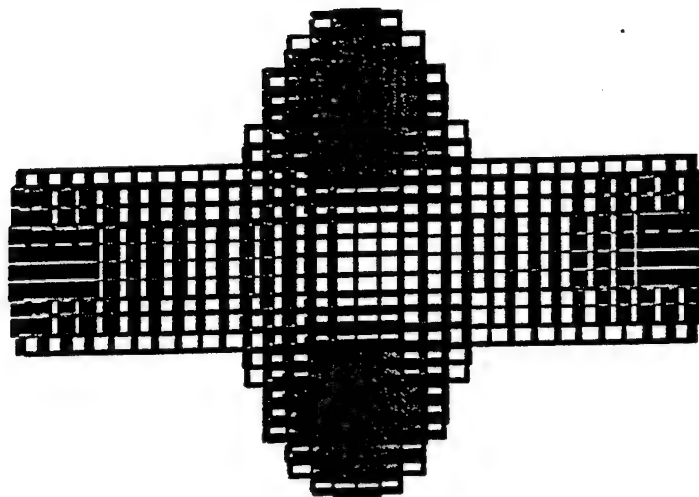


Figure 10. Complex Object - Perfect Conducting Elliptical Shape With a Cylinder Rod Through the Center.

## 7. Description of Algorithm Runs

The Phillips Laboratory FDTD algorithm, as shown in the Appendix, written by Dr John Beggs and developed from Yee's algorithm was modified to obtain the data that are required for this analysis. The program was modified so that the impulse waveform and the pulsed CW could be used as the input source. The objects were also created so that they could be used for targets for the FDTD algorithm. Pseudo codes for the FDTD algorithm which consists of a main program and a post process program are shown below.

### Main Code (temac3d.f)

call setup - initializes equation multipliers and material constitutive parameters

call readmesh - build the mesh by assigning materials to building blocks

Do n = 1, number of time steps

    call efields - advance x, y, and z efield components one time step

    call liao - apply outer radiation boundary conditions. This allows the  
        scattered fields to travel outward without reflecting off the mesh  
        walls.

    call hfields - advance x, y, and z hfield components one time step

    call farfield - integrate efields over surface surrounding scattering object to  
        obtain vector potentials on this surface for each time step

continue

call finishfarfield - transform Cartesian vector potential components on  
    surrounding surface to spherical vector components at each

time step for both scattered and incident fields. Transform  
spherical vector components to far field electric field  
components.

end

Post process code (fzproc.f) - compute radar cross section vs. frequency from data  
output by temac3d.f.

call readin - read in the time domain information from temac3d output such as  
mesh parameters and scattered and incident efield components (far  
zone).

call wrttim - write out the time domain far zone scattered and incident fields into  
separate files.

call cmpfft - perform fast Fourier transforms on efield data

call cmprcs - compute the radar cross section

call wrtrcs - write radar cross section to a data file

end

The FDTD program was processed on a Sun SPARC Station 10 computer. The  
runs for the perfect conducting sphere and square plate with the Gaussian waveform took  
approximately 14 minutes with 1000 time steps. With the impulse waveform and the  
pulsed CW, the runs took approximately one hour because 5000 time steps were used.

## 8. Analysis of Results

To make sure the FDTD algorithm was obtaining the correct results, the code was modified to use a Gaussian input so the results could be compared to the results obtained by Kunz and Luebbers [12] as shown in Figure 11. The incident angle used was  $22.5^\circ$ . The results from the FDTD algorithm used for this analysis, as shown in Figure 12, was very close to the results Kunz and Luebbers obtained, as shown in Figure 11, for a perfect conducting sphere. For Figures 11 and 12, the radar cross section is in dB and not normalized to  $\pi a^2$ .

There are three distinct regions in the radar cross section for a perfect conducting sphere. The Rayleigh region or the low frequency region of the radar cross section varies as the inverse of the fourth power of the wavelength of the incident radiation. In this region, the circumference is less than one wavelength. The second region, the resonance region ( $1 < 2\pi a/\lambda < 10$ ) is the transition region that is characterized by a damped oscillation about the  $\sigma/\pi a^2 = 1$  value when the radar cross section is normalized to  $\pi a^2$  and not in dB. The interference between a specular component and a component due to waves which are exponentially damped as they creep around the shadowed portion of the sphere and circumnavigate it to be relaunched in the direction of the radar receiving antenna causes the oscillation. The optical or high frequency region ( $2\pi a/\lambda > 10$ ), the radar cross section levels down to a value of  $\sigma/\pi a^2 = 1$  when normalized to  $\pi a^2$ , which is the visual projected area of the sphere [14].

The angle of incidence was changed to  $0^\circ$  to compare the radar cross section of the sphere for the impulse waveform and the pulsed CW. In addition, the Gaussian was also

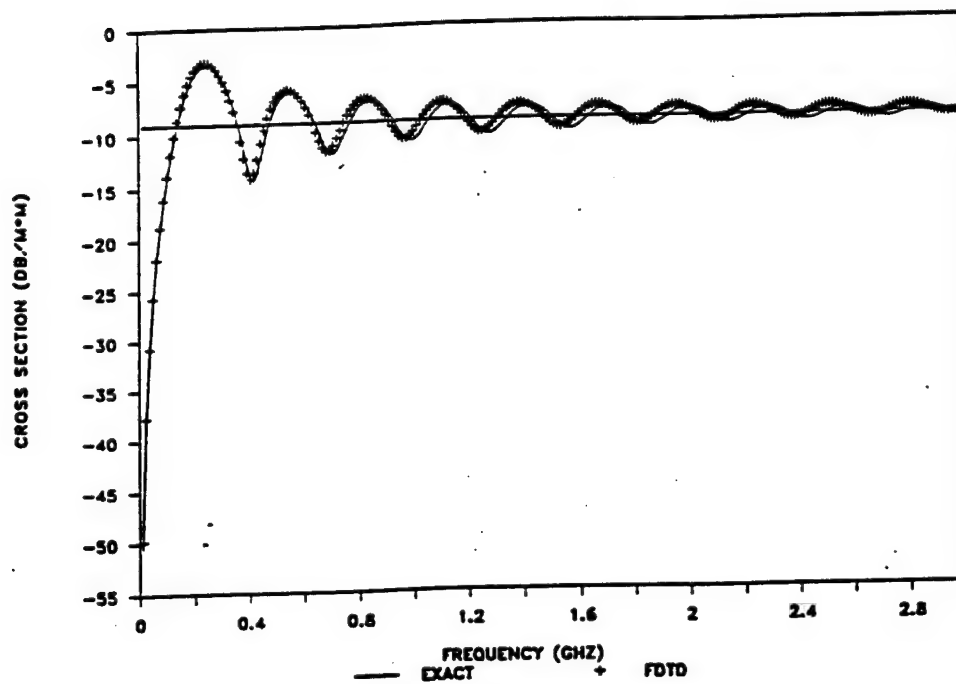


Figure 11. Radar Cross Section of Perfect Conducting Sphere obtained by Kunz and Luebbers [12] for Gaussian input waveform, 1000 time steps, incident angle =  $22.5^\circ$ .

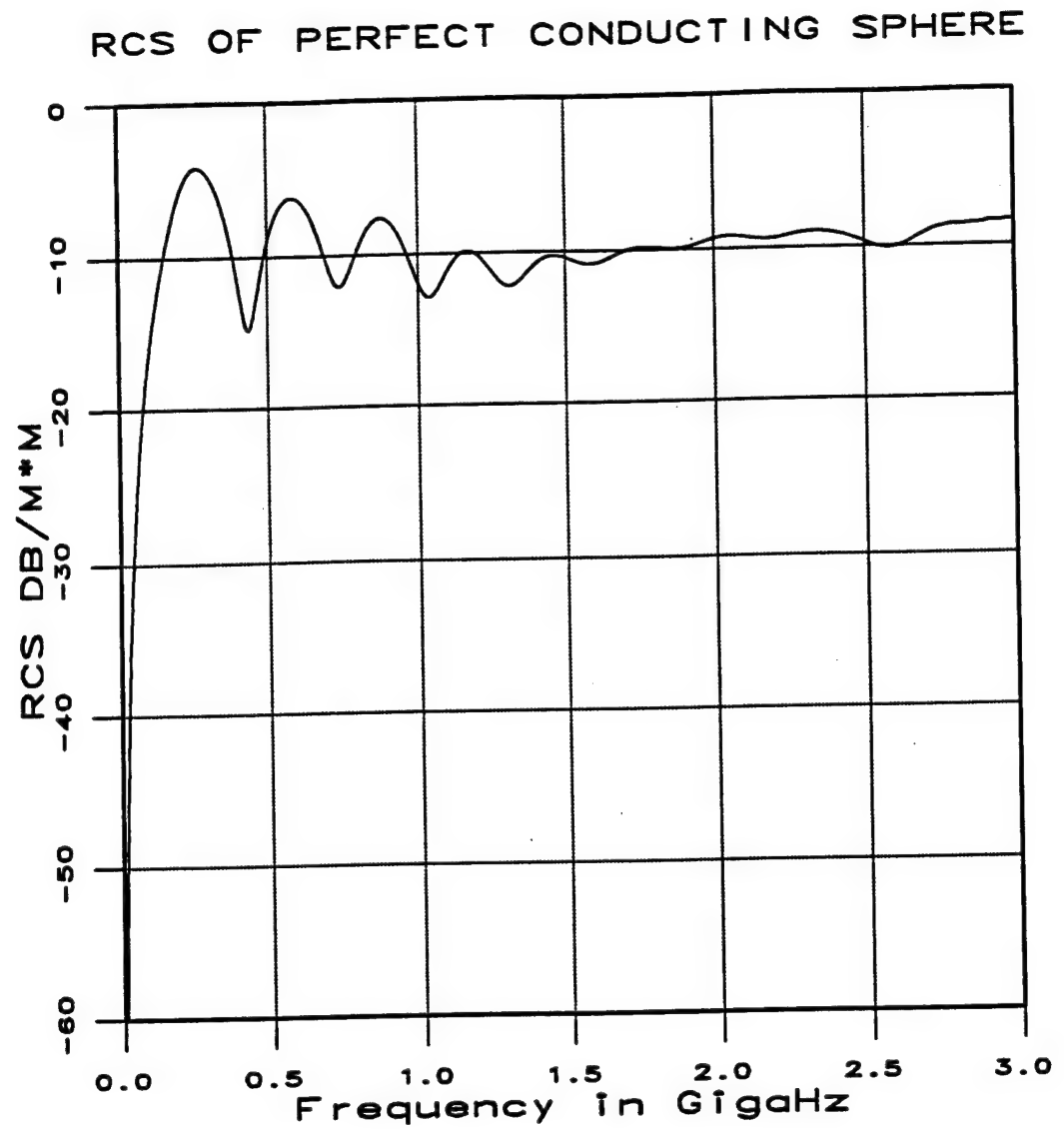


Figure 12. Radar Cross Section of Perfect Conducting Sphere for Gaussian input waveform, 1000 time steps, incident angle =  $22.5^\circ$ .

used as a baseline as shown in Figure 13. Figure 14 shows the radar cross section of the perfect conducting sphere for the impulse waveform. The vertical lines that are displayed on the graph are caused by numerical error and because the amplitude of the input plane wave is nearly zero outside 600 MHz to 1.1 GHz which is the same order of magnitude of the round off error due to the finite differencing (terms of the Taylor series are dropped). Figure 15 shows the pulsed CW radar cross section of the perfect conducting sphere. At 1 GHz the graph matches both the Gaussian and impulse waveforms. However, at all the other frequencies, the data is incorrect due to the input signal of the pulsed CW having zero amplitude at all frequencies except at 1 GHz.

Figure 16 shows the radar cross section of the perfect conducting square plate for a Gaussian input waveform with an incident angle of zero. For a planar surface which is placed perpendicular to the range vector from the radar (i.e., normal incidence) its effective aperture which intercepts the power flux density is equal to its area  $A$ . Because the surface of the square plate is smooth, most of the power is reflected back in the perpendicular direction, with a gain related to the aperture  $A$ , as follows:

$$G = \frac{4\pi A}{\lambda^2}. \quad (111)$$

The effective cross section of a large, smooth plate in the normal direction is given as

$$\sigma = AG = \frac{4\pi A^2}{\lambda^2}, \quad 2\pi A^{1/2} \gg \lambda. \quad (112)$$

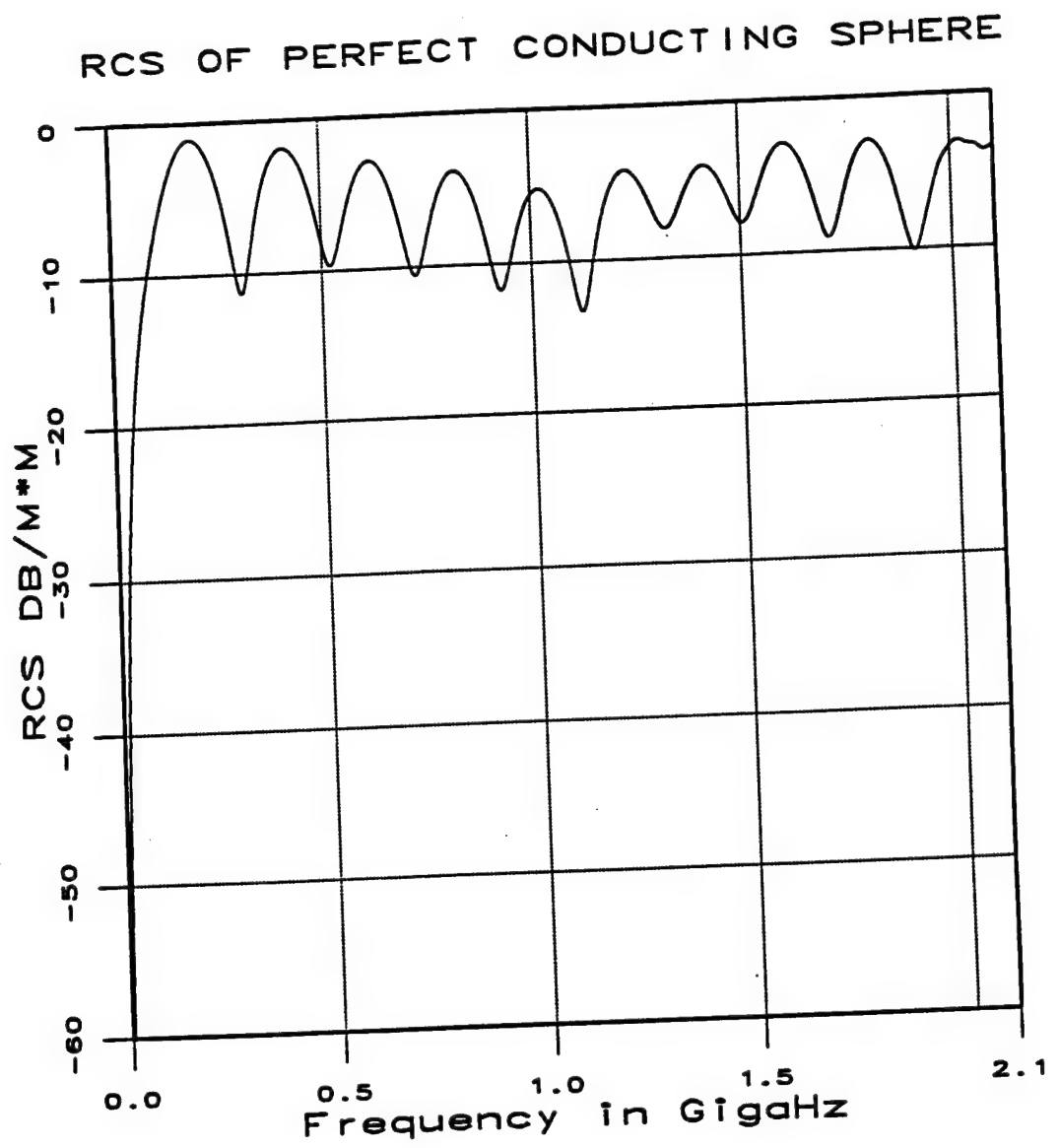


Figure 13. Radar Cross Section of Perfect Conducting Sphere for Gaussian input waveform, 1000 time steps, incident angle =  $0^0$ .

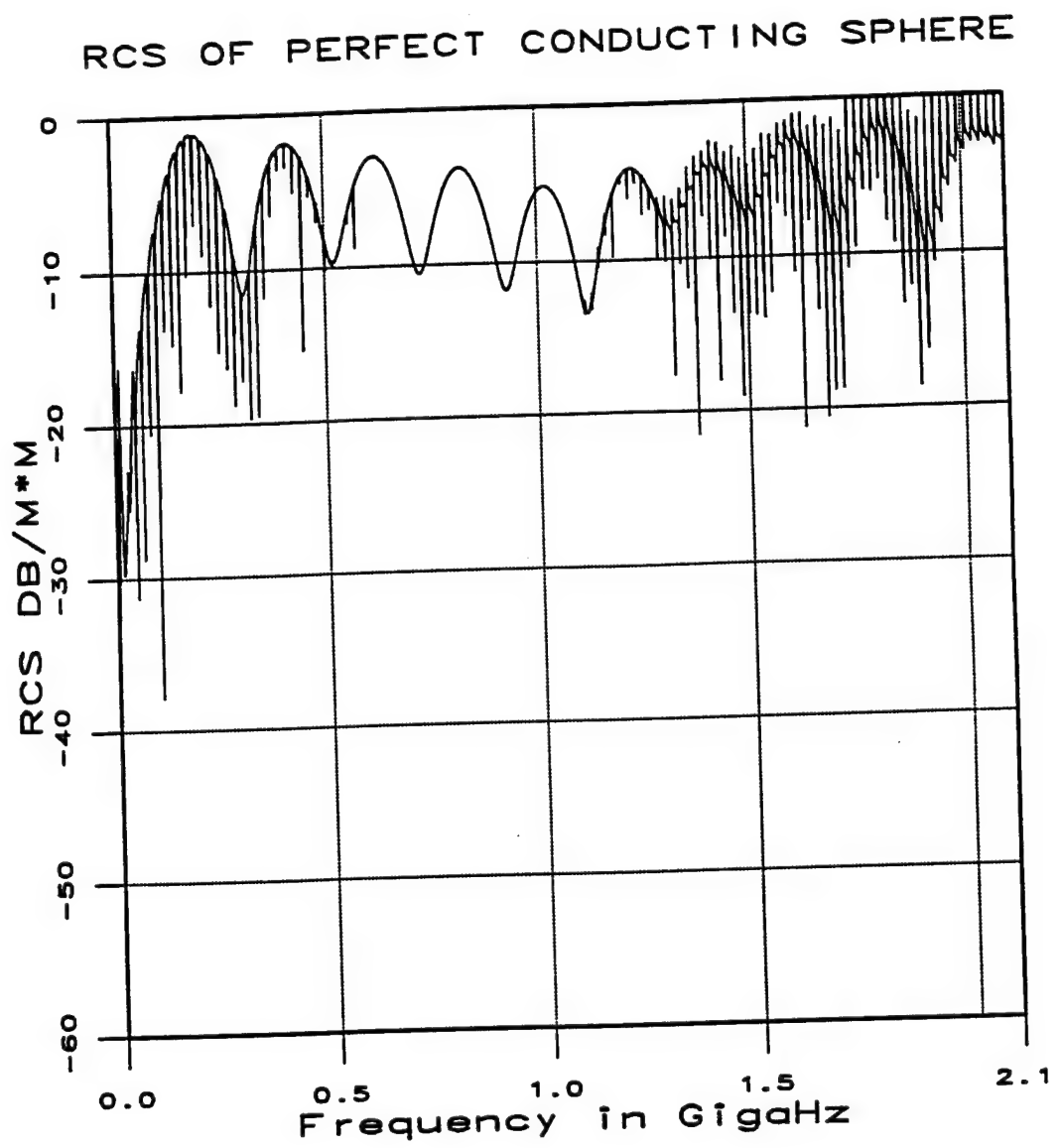


Figure 14. Radar Cross Section of Perfect Conducting Sphere for Impulse Input Waveform, 5000 Time Steps, Incident Angle =  $0^0$ .

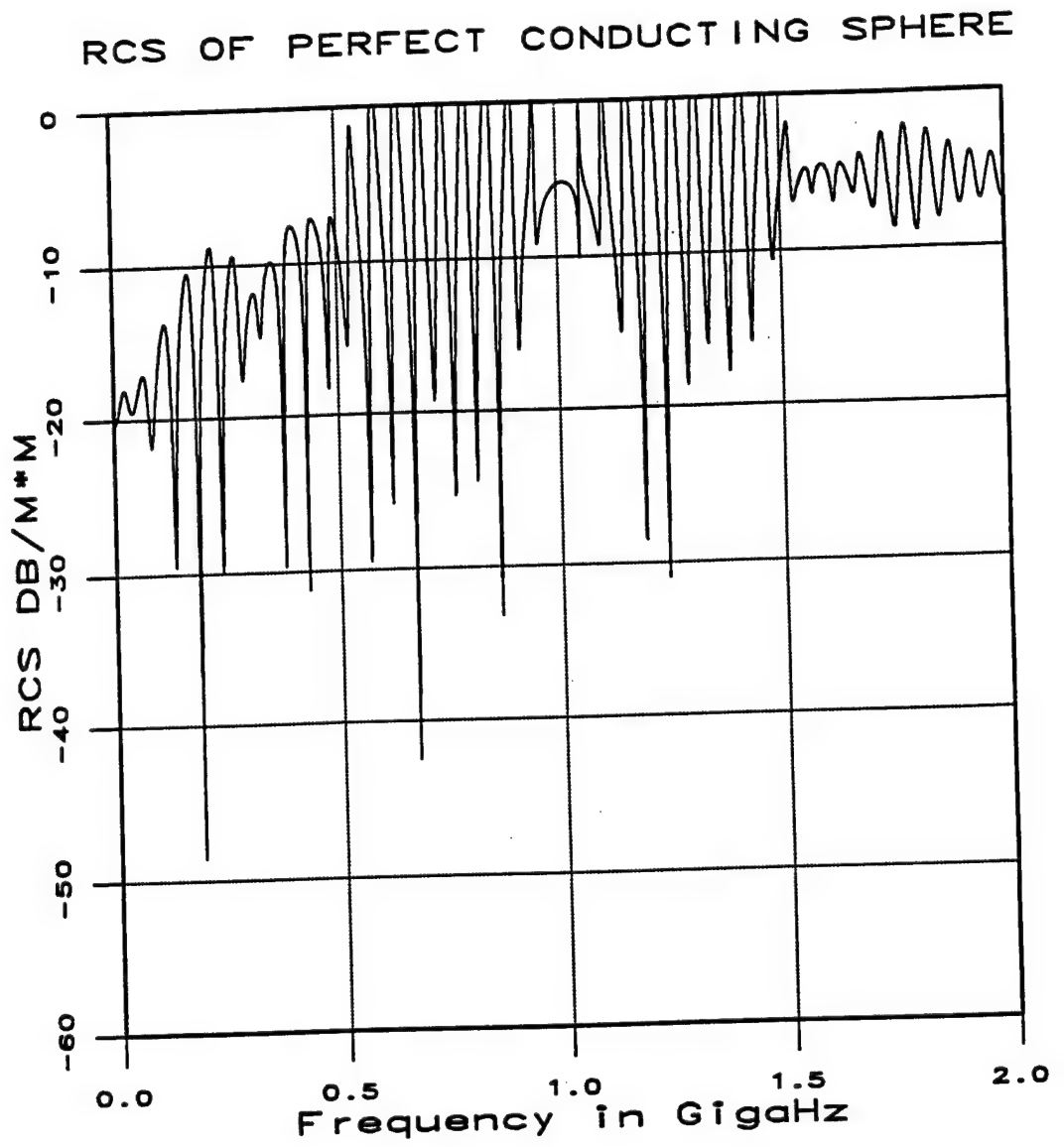


Figure 15. Radar Cross Section of Perfect Conducting Sphere for Pulsed CW Input Waveform, 5000 time steps, Incident Angle =  $0^{\circ}$ .

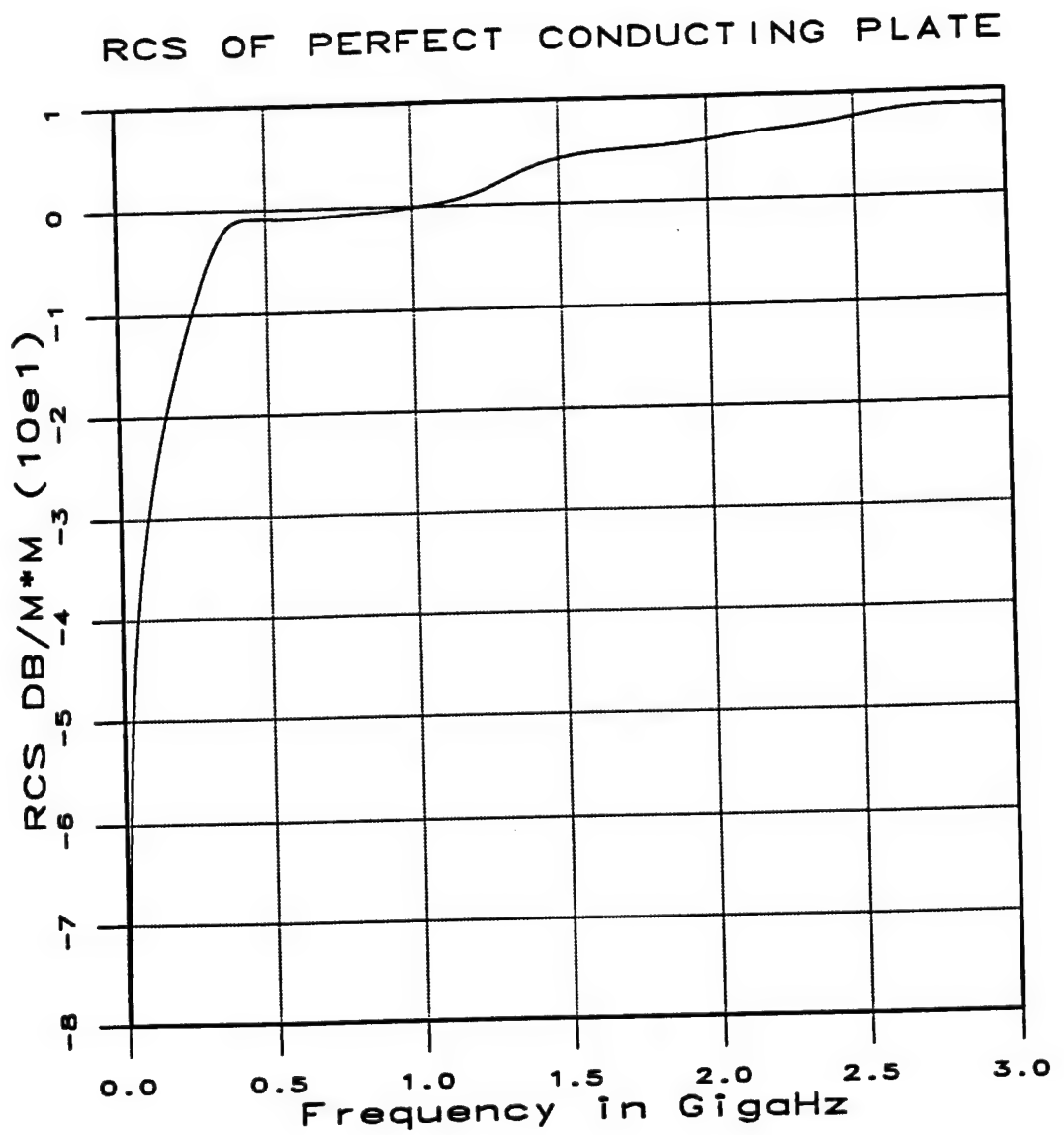


Figure 16. Radar Cross Section of Perfect Conducting Square Plate for Gaussian input waveform, 5000 time steps, incident angle =  $0^0$ .

Equation (112) implies that  $\sigma$  of a large plane is much larger than the area  $A$  of that plane, but only at a normal incidence. At other than normal incidence the dependence of  $\sigma$  on the angle of incidence (measured from the normal), for a square flat plate is given as

$$\sigma = \frac{4\pi a^4}{\lambda^2} \left[ \frac{\sin(ka \sin\theta)}{(ka \sin\theta)} \right]^2 \quad (113)$$

where  $a$  is the length of the side [8]. When the impulse input waveform is used, as shown in Figure 17, the graph matches the Gaussian input waveform in the 600 MHz to 1.1 GHz frequency range. Also, the pulsed CW waveform as shown in Figure 18 is in agreement at the operating frequency of 1 GHz.

Next, the incident angle was changed from normal incidence to  $22.5^\circ$  from the normal in order to observe the difference in polarization for both the impulse and pulsed CW input waveforms. Figure 19 shows the radar cross section of the perfect conducting square plate for the impulse input waveform with a horizontal polarization and an incident angle of  $22.5^\circ$ . The polarization was changed to vertical and the radar cross section is shown in Figure 20. It follows from these graphs, the radar cross section is different.

For a complete description of the interaction of the incident wave and the target is given by the polarization scattering matrix ( $S$ ) which relates  $E_s$  and  $E_i$  component-by-component. Since  $E$  can be decomposed into two independent directions or polarizations,  $S$  is the four element matrix

$$\begin{bmatrix} E_{s1} \\ E_{s2} \end{bmatrix} = \begin{bmatrix} S_{11} & S_{12} \\ S_{21} & S_{22} \end{bmatrix} \begin{bmatrix} E_{i1} \\ E_{i2} \end{bmatrix} \quad (114)$$

# RCS OF PERFECT CONDUCTING SPHERE

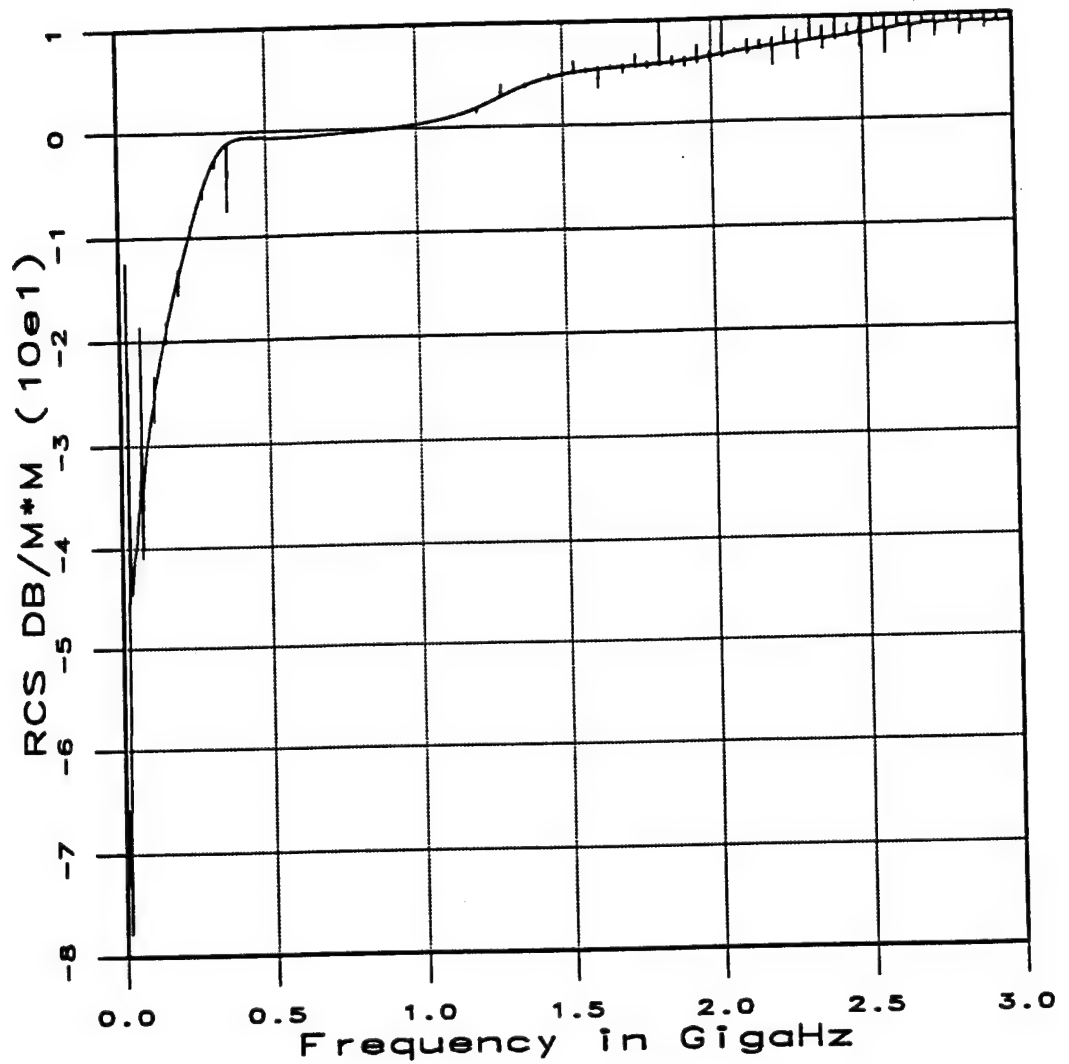


Figure 17. Radar Cross Section of Perfect Conducting Square Plate for Impulse Input Waveform, 5000 Time Steps, Incident Angle =  $0^{\circ}$ .

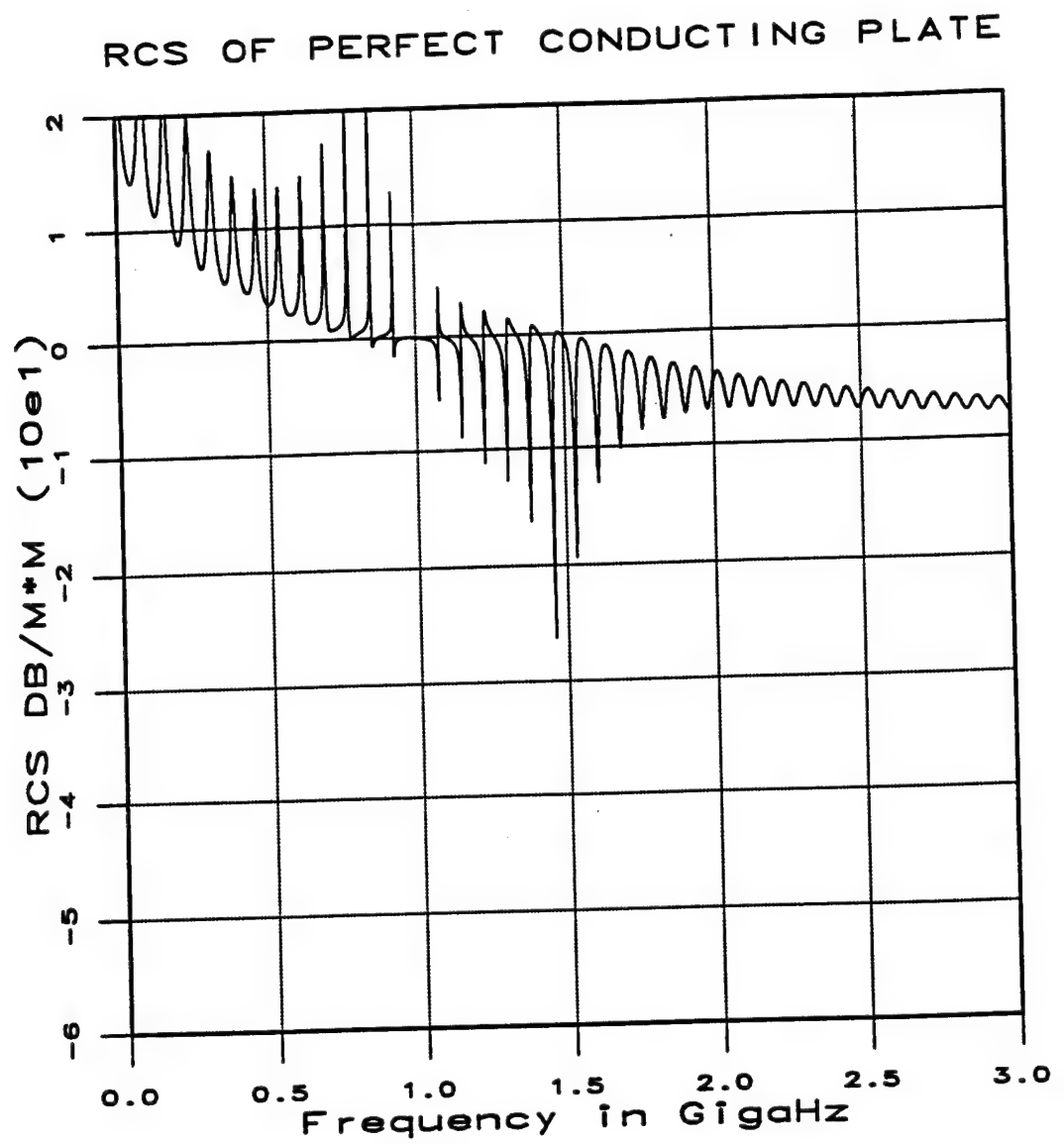


Figure 18. Radar Cross Section of Perfect Conducting Square Plate for Pulsed CW Input Waveform, 5000 Time Steps, Incident Angle =  $0^{\circ}$ .

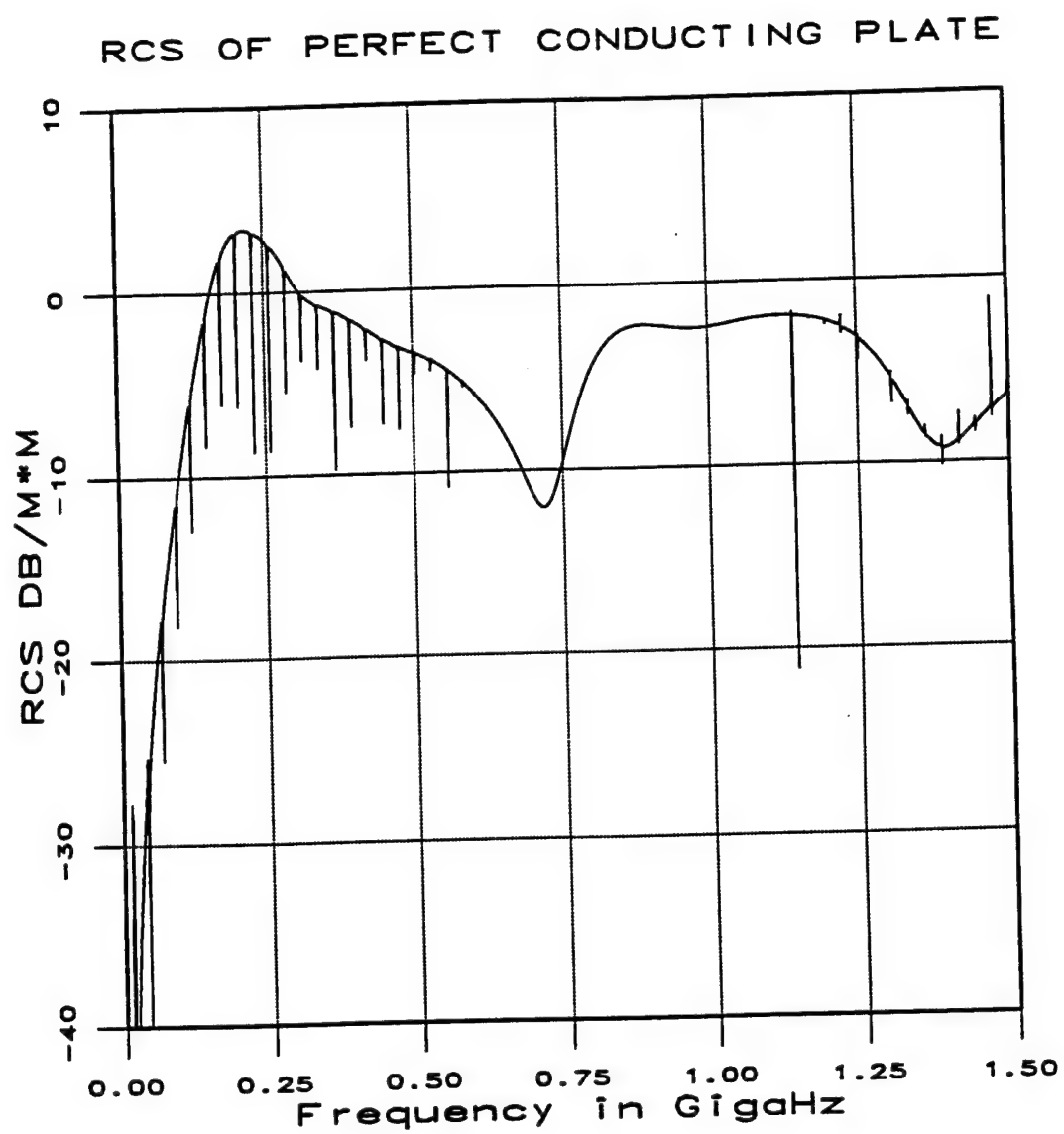


Figure 19. Radar Cross Section of Perfect Conducting Square Plate for Impulse Input Waveform, Horizontal Polarization, Incident Angle =  $22.5^\circ$ .

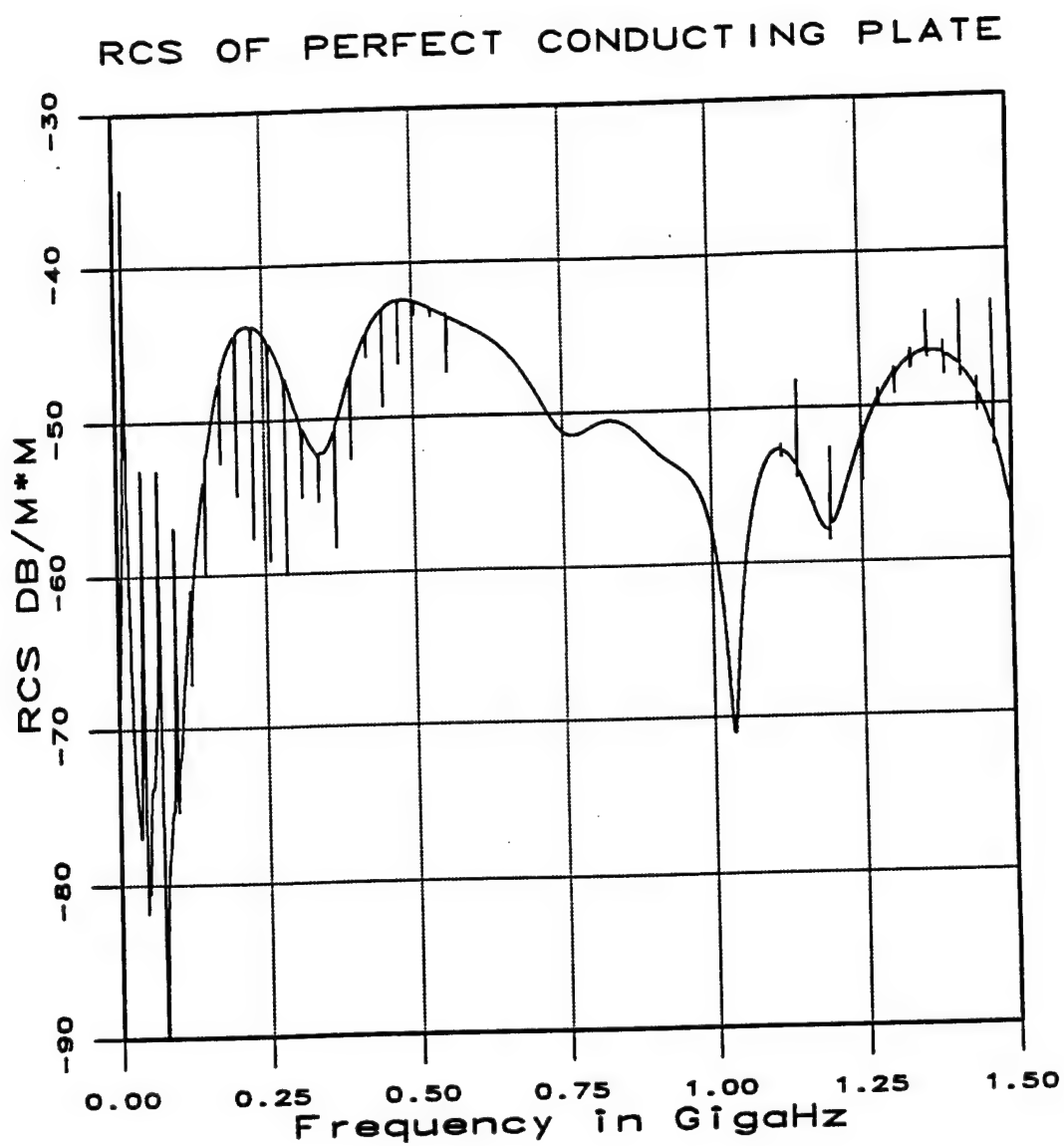


Figure 20. Radar Cross Section of Perfect Conducting Square Plate for Impulse Input Waveform, Vertical Polarization, Incident Angle =  $22.5^\circ$ .

The components of  $S$  are related to target radar cross section by

$$S_{jk} = (4\pi r^2)^{-1/2} \sqrt{\sigma_{jk}} \quad (115)$$

where  $\sqrt{\sigma}$  is a complex number that contains phase as well as amplitude information [6].

The radar cross section for the pulsed CW input waveform is also different for horizontal and vertical polarizations. Although the only information we get for the pulsed CW is at 1 GHz, the amplitude for the horizontal polarization, as shown in Figure 21, is much higher than that for vertical polarization as shown in Figure 22.

Next, a complex object, consisting of an ellipsoid with a cylinder, was used as a target. This object consists of a number of different surfaces. Therefore, the radar cross section is very difficult to predict. Figure 23 shows the radar cross section of the perfect conducting ellipsoid with a cylinder for the impulse input waveform with a horizontal polarization and an incident angle of  $22.5^\circ$  from the normal. The polarization was changed to vertical and the radar cross section is shown in Figure 24. Once again it is shown that the radar cross section is very different between the two polarizations.

Figure 25 shows the radar cross section for the pulsed CW input waveform with a horizontal polarization and an incident angle of  $22.5^\circ$  from the normal. The vertical polarization radar cross section for the pulsed CW input waveform, as shown in Figure 26, shows a much lower amplitude at 1 GHz than for horizontal polarization.

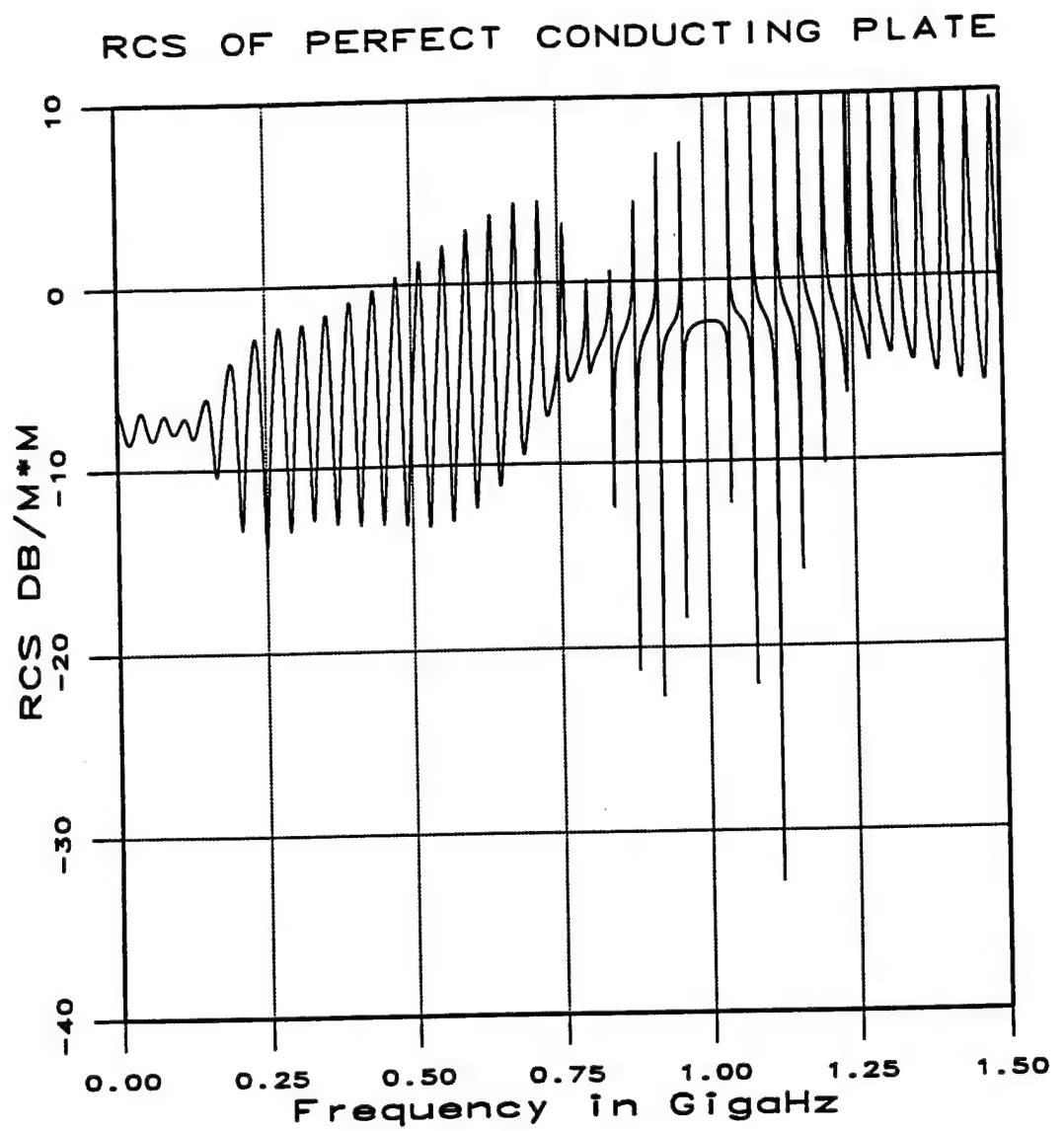


Figure 21. Radar Cross Section of Perfect Conducting Square Plate for Pulsed CW Input Waveform, Horizontal Polarization, Incident Angle =  $22.5^{\circ}$ .

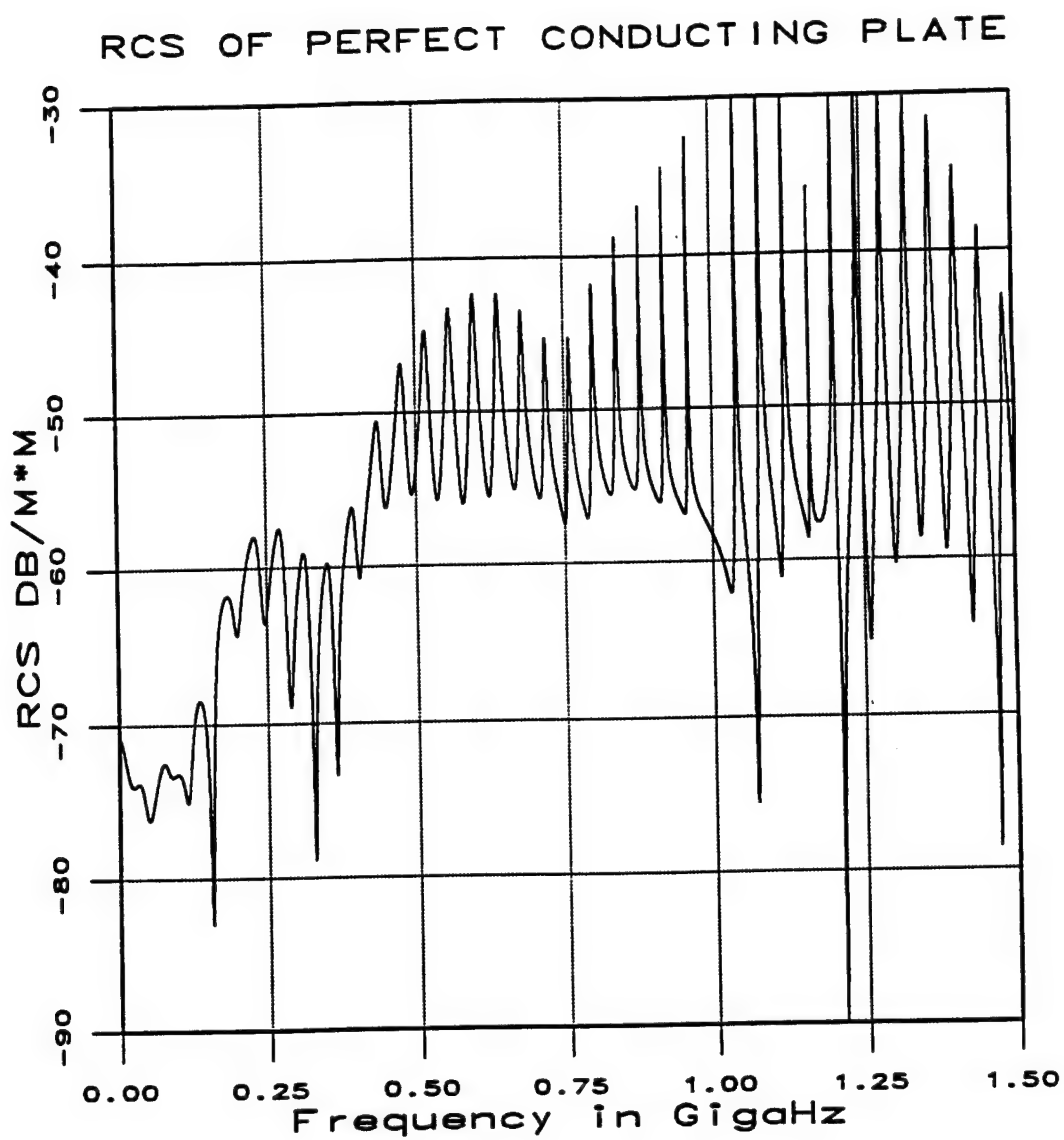


Figure 22. Radar Cross Section of Perfect Conducting Square Plate for Pulsed CW Input Waveform, Vertical Polarization, Incident Angle =  $22.5^\circ$ .

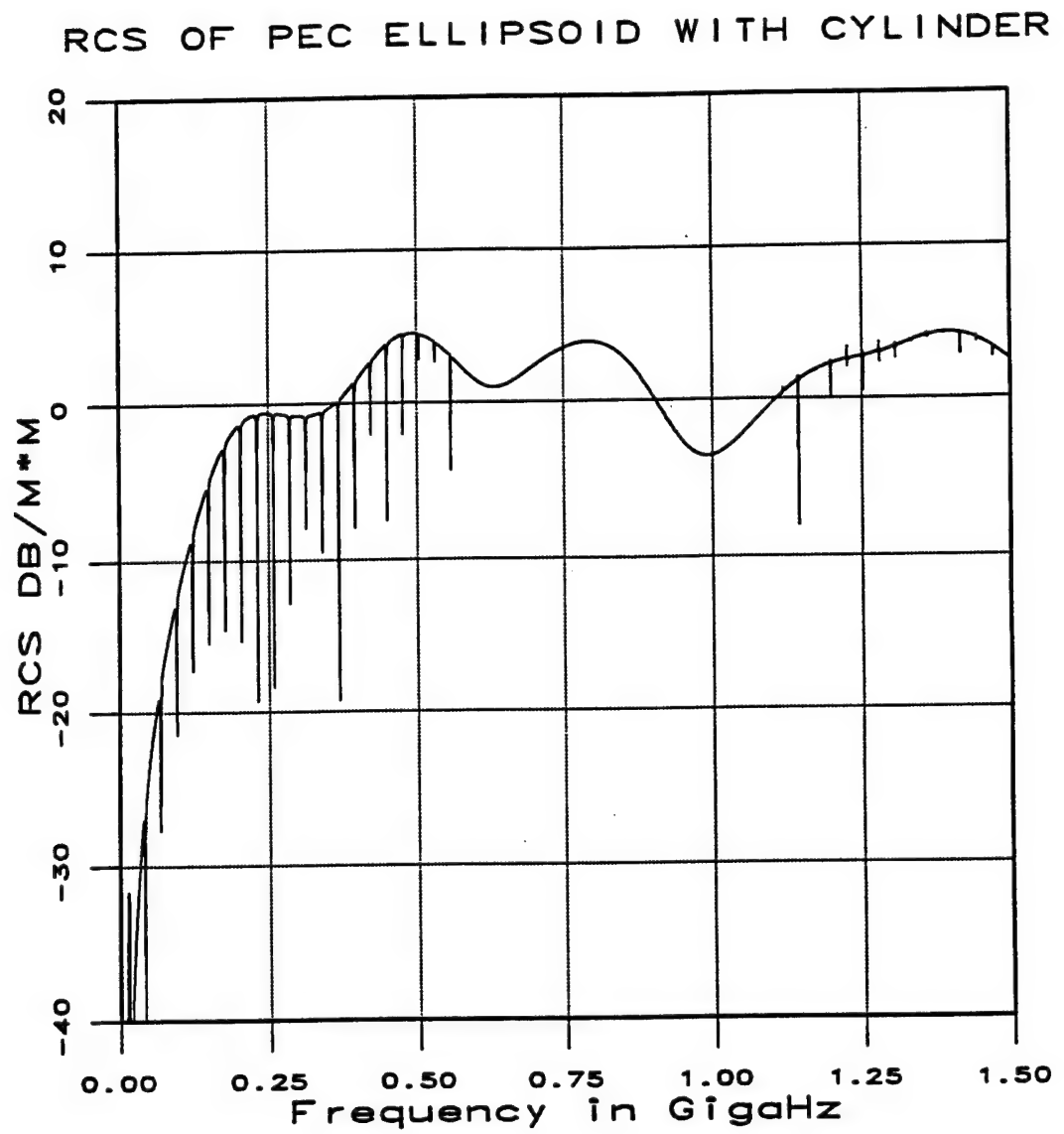


Figure 23. Radar Cross Section of Perfect Conducting Ellipsoid with a Cylinder for Impulse Input Waveform, Horizontal Polarization, Incident Angle =  $22.5^\circ$ .

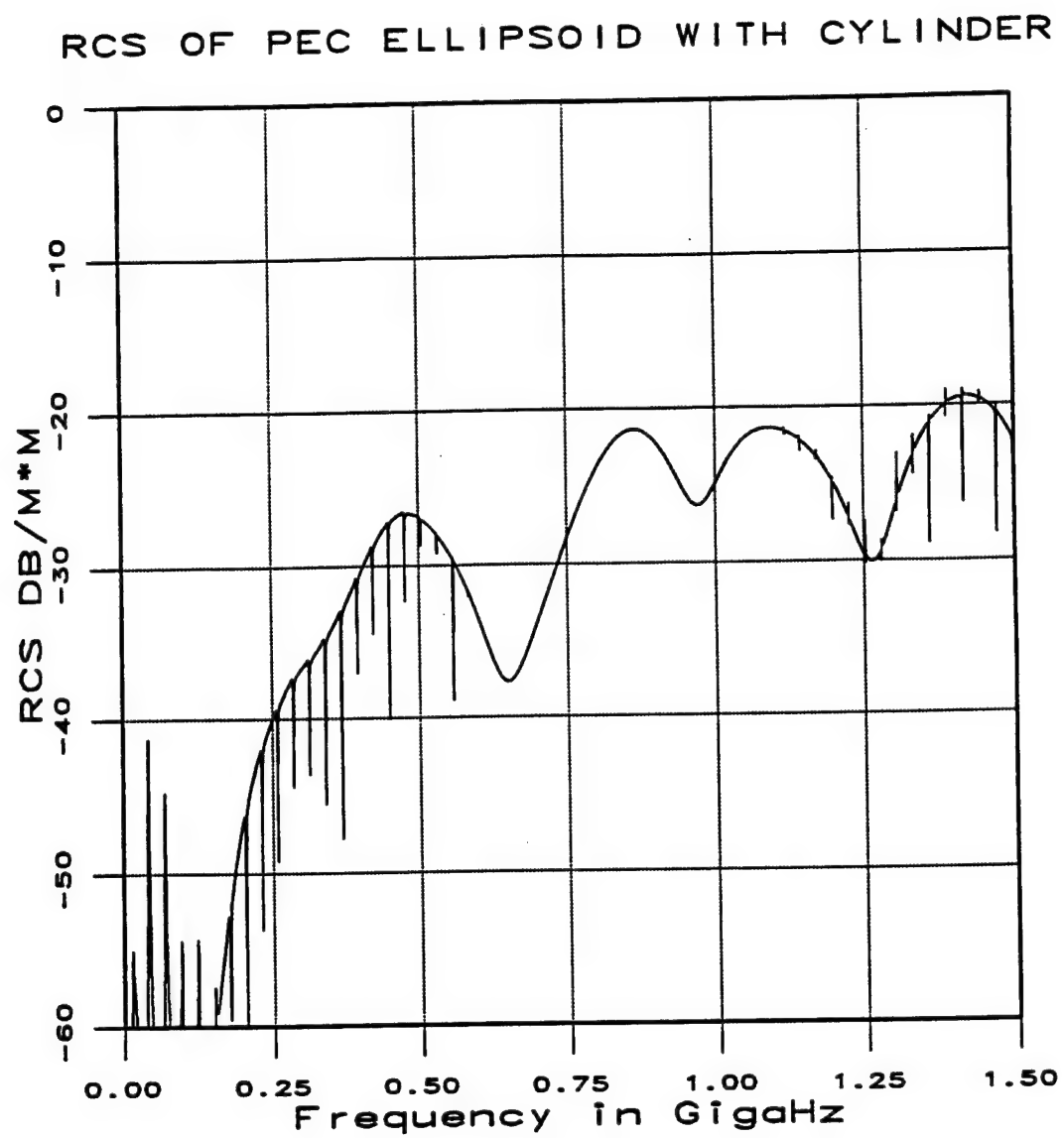


Figure 24. Radar Cross Section of Perfect Conducting Ellipsoid with a Cylinder for Impulse Input Waveform, Vertical Polarization, Incident Angle =  $22.5^\circ$ .

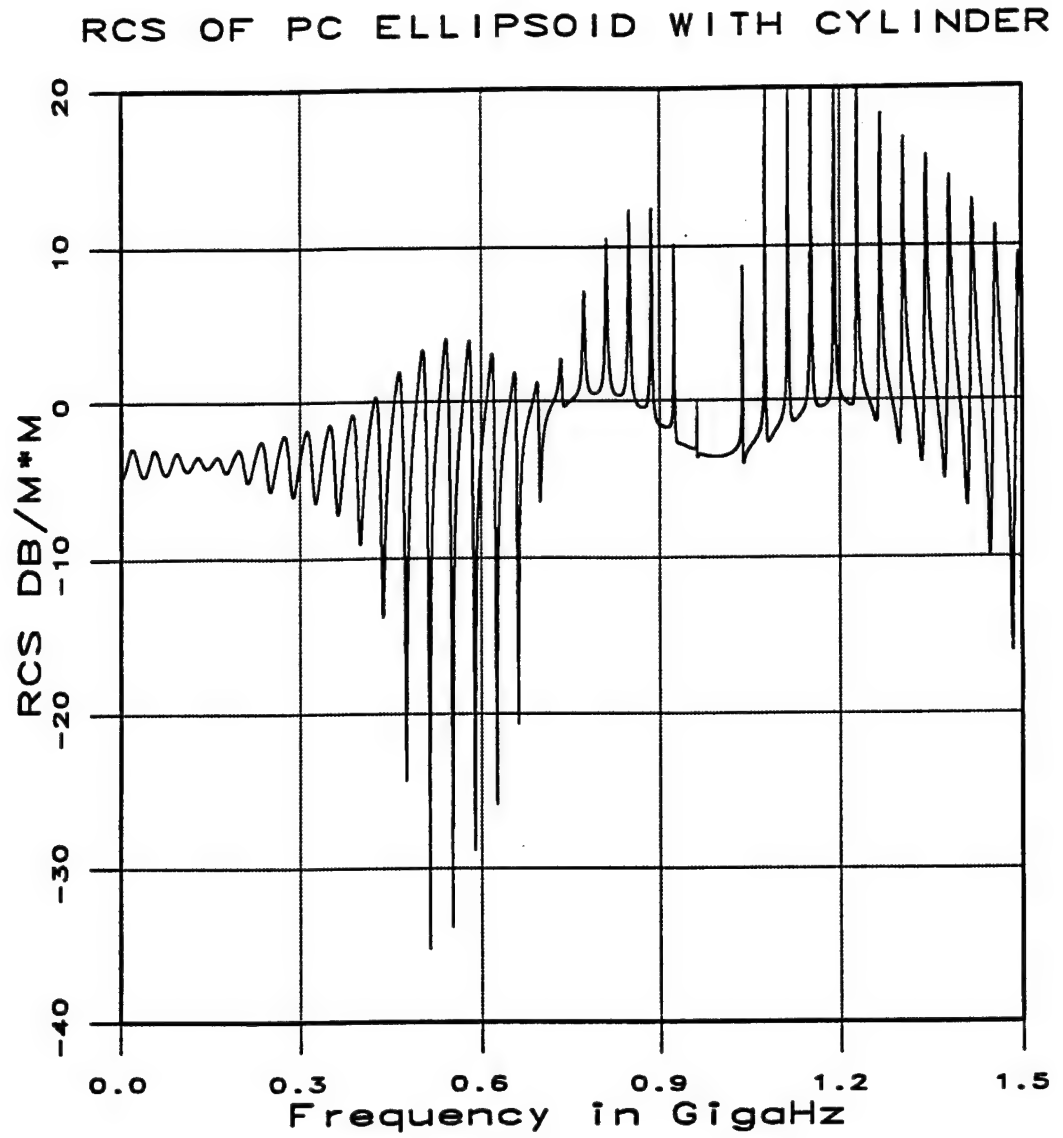


Figure 25. Radar Cross Section of Perfect Conducting Ellipsoid with a Cylinder for Pulsed CW Input Waveform, Horizontal Polarization, Incident Angle =  $22.5^\circ$ .

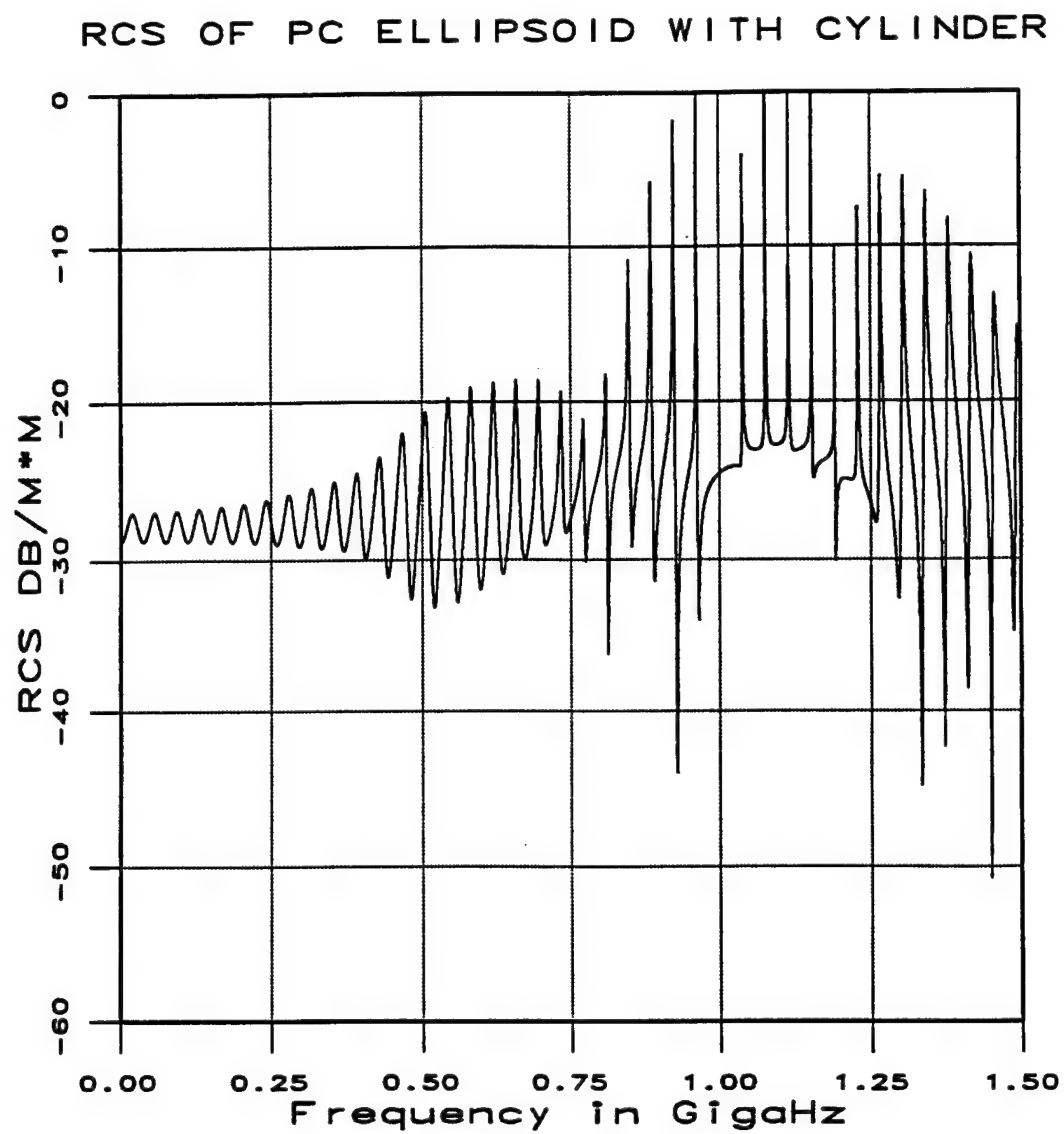


Figure 26. Radar Cross Section of Perfect Conducting Ellipsoid with a Cylinder for Pulsed CW Input Waveform, Vertical Polarization, Incident Angle =  $22.5^\circ$ .

## 9. Conclusions

The major advantage of using an impulse radar is that there will be more body resonances from an object because of the larger frequency band. Therefore, the radar cross section will be very close to an ideal Gaussian in the frequency band of operation. The pulsed CW waveform will only be exact at the one frequency of operation. Therefore, with an impulse waveform, detectability and identification/resolution can be greatly improved when compared to a pulsed CW radar.

The radar cross section data from this analysis demonstrated that the impulse waveform provides more target information than from the pulsed CW waveform. The resolution of an object would be finer with an impulse waveform than with a pulsed CW waveform, as depicted in Figure 27. Figure 27 (a) shows the resolution of an object using a pulsed CW radar. At the exact frequency of the body resonance ( $\omega_c$ ), the resolution of the object is fine. If the pulsed CW radar does not operate at the exact body resonance ( $\omega_c + \Delta\omega$  or  $\omega_c - \Delta\omega$ ), the resolution of the object will not be fine but will be obscure. The poor resolution makes it very difficult to identify the object. With an impulse radar which has a large frequency spectrum, the probability of exciting a body resonance of an object is very high. This allows for finer resolution and improves identification of an object as shown in Figure 27 (b). This can be very important in target recognition, especially for detection of underground objects such as mines. By using a modeling code such as FDTD, a library of signatures for individual targets, and for specific classes of targets can be established. This would enable one to compare the library of stored signatures to a measured signature for identifying a target.

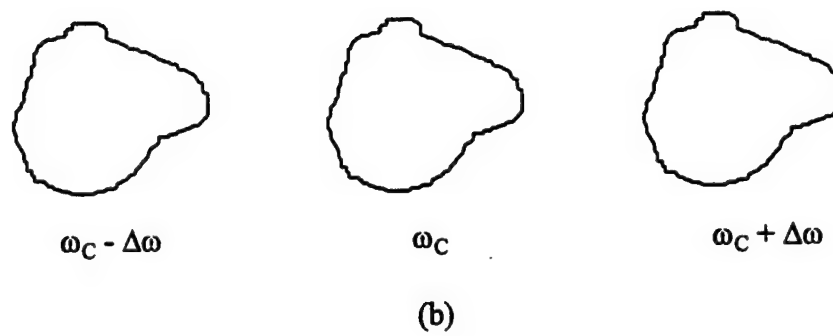
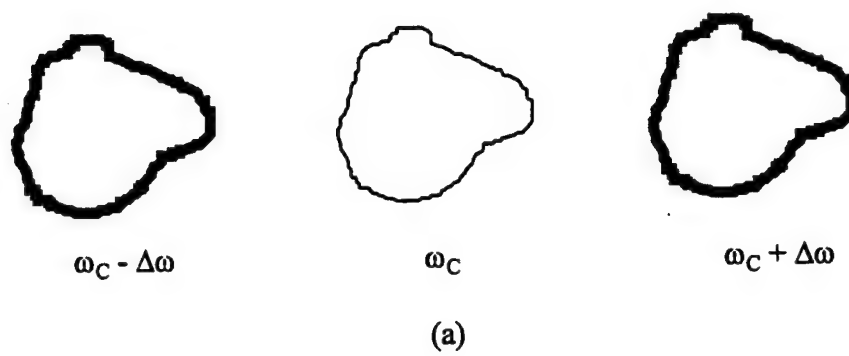


Figure 27. Example of Resolution for an Object Using (a) Pulsed CW and (b) Impulse Waveform.

The advantages of the FDTD code used in this analysis is its ability to work with a wide range of frequencies, stimuli, objects, environments, response locations, and computers. To this list can be added the advantage of computational efficiency for large problems in comparison with other techniques such as the moment of methods, especially when broadband results are required. Its accuracy, using a sufficient number of cells, can be made as high as desired. Conversely, engineering estimates of a few decibels' accuracy can be made with a few cells [12].

An impulse radar does not only have a finer resolution than a pulsed CW radar, but also provides low clutter. This significantly improves the target detectability in strong range distributed clutter. Moreover, since the impulse is very short, it is ideal for target range determination, using time delay techniques. Once the target is detected, the determination of its range involves measuring the time for the signal to travel from the source to the target and then scatter back to the receiver. A pulse train can easily be tagged by an exaggerated or abbreviated duty cycle to eliminate pulse ambiguities. The target azimuth and elevation would then be determined by conventional pointing and tracking techniques.

Additional signal processing can also be used to enhance the signature(s), extract the values of the parameter set of interest, and reduce noise which can add to the advantages of temporal resolution and large spectral information that an impulse radar provides. For example, the Prony method may be used to extract poles, which provides a means of target identification using target natural resonance responses as a type of target signature. A recommended continuation of this thesis would not only consist of

additional signal processing, but also would include a comparison of the analysis with actual impulse radar experimental data.

## References

1. L. Cai, H. Wang, and C. Tsao, "Moving Target Detection Performance Potential of UWB Radars," Proc. of the SPIE Ultrahigh Resolution Radar, vol. 1875, pp. 2-13, 1993.
2. D. R. Wehner, High Resolution Radar, Artech House, Norwood, Mass., 1987.
3. A. W. Rihaczek, Principles of High-Resolution Radar, McGraw-Hill, New York, New York, 1969.
4. R. L. Hutchins, and C. B. Wallace, "Effects of Sampling and Propagation Loss on Ultrawideband Synthetic Aperture Radar," Proc. of the SPIE Ultrawideband Radar, vol. 1631, pp. 36-42, 1992.
5. D. P. Meyer and H. A. Mayer, Radar Target Detection Handbook of Theory and Practice, Academic Press, Orlando, Florida, 1973.
6. E. F. Knott, J. F. Shaeffer, and M. T. Tuley, Radar Cross Section, Artech House, Norwood, Mass., 1985.
7. M. I. Skolnik, Introduction to Radar Systems, McGraw-Hill, New York, New York, 1962.
8. N. Levanon, Radar Principles, John Wiley & Sons, New York, New York, 1988.
9. K. A. Shubert, and G. T. Ruck, "Canonical Representation of the Radar Range Equation in the Time Domain," Proc. of the SPIE Ultrawideband Radar, vol. 1631, pp. 2-12, 1992.
10. J. M. Ralston, "System Analysis of Ultra-Wideband Instrumentation Radars: Impulse vs. Stepped-Chirp Approaches," Proc. of the SPIE Ultrahigh Resolution Radar, vol. 1875, pp. 114-123, 1993.
11. J. W. Crispin, and K. M. Siegel, Methods of Radar Cross-Section Analysis, Academic Press, New York, New York, 1968.
12. K. S. Kunz, and R. J. Luebbers, The Finite Difference Time Domain Method for Electromagnetics, CRC Press, Boca Raton, Florida, 1993.
13. K. S. Yee, "Numerical Solution of Initial Boundary Value Problems Involving Maxwell's Equations in isotropic Media," IEEE Trans. Antenna Prop., 14(3), Pg 302, 1966.

14. V. C. Martins, et al, Signal Processing Study of Impulse Excited Objects in Space, Rome Air Development Center, Giffiss Air Force Base, New York, 1973.

## **Appendix**

```

C
C      PROGRAM TEMAC
C
C      Phillips Laboratory TEMAC3D code--Temporal
C      ElectroMagnetic Analysis Code.
C      This code is a Finite-Difference Time-Domain
C      Electromagnetic Analysis code based upon the
C      Yee algorithm [1]. This version of TEMAC3D
C      has the following capabilities:
C
C      1.) Free space material
C      2.) Perfect electrical conductor material
C      3.) Lossy dielectric material
C      4.) Lossy magnetic material
C      5.) Dispersive dielectric material (not yet)
C      6.) Dispersive magnetic material (not yet)
C      7.) Surface impedance (not yet)
C      8.) Nonlinear materials (not yet)
C      9.) Nonlinear lumped elements (not yet)
C      10.) Thin slots (not yet)
C      11.) Thin wires (not yet)
C      12.) Lumped circuit elements (not yet)
C      13.) Thin sheets (not yet)
C      14.) Liao Outer Radiation Boundary Condition (ORBC)
C      15.) Near to far-field transformation
C      16.) Point source or plane wave source
C      17.) Point sensor or slice sensor
C
C      For additional information contact:
C
C      Dr. John H. Beggs
C      PL/WSR
C      3550 Aberdeen Ave. SE
C      Kirtland AFB, NM 87117-5776
C      (505) 846-4482
C      DSN: 246-4482
C      FAX: (505) 846-0417
C      E-mail: jbeggs@chili.plk.af.mil
C
C      #####
C      Now begins the main portion of the program.
C      #####
C
C      INCLUDE 'main.h'
C
C      real dtime, etime, timer(2)
C      external dtime, etime
C
C      Zero variables and arrays
C
C      This subroutine call eliminates any appreciable time penalties
C      associated with underflowing
C
C      call abrupt_underflow()

```

```

c      call nonstandard_arithmetic()

      CALL ZERO

c
c      Compute the constants used
c
c      CALL SETCON

c
c      Set up the incident field, sensors, constant multipliers
c      defaults and user definitions
c
c      CALL SETUP

c
c      Read the mesh file if a geometry is desired
c
c      IF (.NOT.freesp) CALL READMS

c
c      Initialize the far field transformation if desired
c
c      IF (ffldon) CALL INITFF

c
c      Initialize the Liao ORBC
c
c      CALL STLIAO

c
c      The main loop for field solution is next.  The organization
c      is as follows:
c
c      1.) Update Ex field components
c      2.) Update Ey field components
c      3.) Update Ez field components
c      4.) Update time by 1/2 step
c      5.) Update any electric field point sources
c      6.) Update Hx field components
c      7.) Update Hy field components
c      8.) Update Hz field components
c      9.) Update time by 1/2 step
c      10.) Save point or slice sensor data
c      11.) Update far-field vector potentials
c
c      DO 10 n=1,tsteps
c
c          IF (mod(n,10).EQ.0) THEN
c              OPEN (UNIT=99,FILE='howfar',access='append')
c              if(n.eq.10)rewind 99
c              write(99,*)'etime=',etime(timer)
c              write(99,*)'dtime=',dtime(timer)
c              WRITE (99,*) 'At time step ',n,' of ',tsteps
c              WRITE (*,*) 'At time step ',n,' of ',tsteps
c              CLOSE (UNIT=99)
c          ENDIF

c
c      Update Ex field components
c
c      CALL UPDEXS

c
c      Update Ey field components
c

```

```

CALL UPDEYS
C
C Update Ez field components
C
CALL UPDEZS
C
C Update any point source functions
C
C IF (pntsrc) CALL PSRCE
C
C Apply Liao Outer Radiation Boundary Condition
C
CALL LIAO
C
C Advance time by 1/2 step
C
time=time+delto2
C
C Update Hx field components
C
CALL UPDHXS
C
C Update Hy field components
C
CALL UPDHYS
C
C Update Hz field components
C
CALL UPDHZS
C
C Advance time by 1/2 step
C
time=time+delto2
C
C Save sensor data if desired
C
C IF ((npoint .ge. 1).OR.(nslice .ge. 1)) CALL SENSOR
C
C Update far field vector potentials if desired
C
C IF (ffldon) CALL FARFLD
C
10 CONTINUE
C
C Finish the far field transformation by transforming
C the Cartesian vector potentials to spherical electric
C far field components
C
C IF (ffldon) CALL FINFF
C
C End of program
C
STOP
END

```

c234567

```
c
      SUBROUTINE ZERO
c
      INCLUDE 'main.h'
c
      This subroutine zeros all of the variables and arrays
      used during the TEMAC execution.
c
c*****
c
c      Local variable dictionary
c
c      i=cell coordinate number in x direction
c      j=cell coordinate number in y direction
c      k=cell coordinate number in z direction
c      l=loop counter over order number for zeroing arrays used
c        in the Liao ORBC
c      m=loop counter over time bins for zeroing far field
c        vector potential array
c
c*****
c
      INTEGER i,j,k,l
      INTEGER m
      time=0.0
      delt=0.0
      dtinv=0.0
      costh=0.0
      sinth=0.0
      cosphi=0.0
      sinphi=0.0
      phipol=0.0
      eamplx=0.0
      eamply=0.0
      eamplz=0.0
      hamplx=0.0
      hamply=0.0
      hamplz=0.0
      samplx=0.0
      samply=0.0
      samplz=0.0
      cosa=0.0
      cosb=0.0
      cosg=0.0
      toff=0.0
      tau0=0.0
      nx1=nx-1
      ny1=ny-1
      nz1=nz-1
      DO 30 k=1,nz
        DO 20 j=1,ny
          DO 10 i=1,nx
            exscat(i,j,k)=0.0
            ey scat(i,j,k)=0.0
            ezscat(i,j,k)=0.0
            hxscat(i,j,k)=0.0
            hyscat(i,j,k)=0.0
          
```

```

        hzscat(i,j,k)=0.0
        id1(i,j,k)=0
        id2(i,j,k)=0
        id3(i,j,k)=0
        id4(i,j,k)=0
        id5(i,j,k)=0
        id6(i,j,k)=0
10      CONTINUE
20      CONTINUE
30      CONTINUE
        DO 70 k=1,nz1
            DO 60 j=1,ny1
                DO 50 l=1,order
                    DO 40 i=1,2
                        eybakx(i,l,j,k)=0.0
                        ezbakx(i,l,j,k)=0.0
40          CONTINUE
50          CONTINUE
60          CONTINUE
70          CONTINUE
            DO 110 k=1,nz1
                DO 100 i=1,nx1
                    DO 90 l=1,order
                        DO 80 j=1,2
                            exbakx(j,l,i,k)=0.0
                            ezbakx(j,l,i,k)=0.0
80          CONTINUE
90          CONTINUE
100         CONTINUE
110        CONTINUE
            DO 150 j=1,ny1
                DO 140 i=1,nx1
                    DO 130 l=1,order
                        DO 120 k=1,2
                            exbakz(k,l,i,j)=0.0
                            eybakz(k,l,i,j)=0.0
120        CONTINUE
130        CONTINUE
140        CONTINUE
150        CONTINUE
            DO 170 m=1,mmax
                DO 160 i=1,6
                    uandw(i,m)=0.0
160        CONTINUE
170        CONTINUE
            DO 190 k=1,nz-9
                DO 180 j=1,ny-9
                    tretx(j,k,1)=0.0
                    tretx(j,k,2)=0.0
180        CONTINUE
190        CONTINUE
            DO 210 k=1,nz-9
                DO 200 i=1,nx-9
                    treti(i,k,1)=0.0
                    treti(i,k,2)=0.0
200        CONTINUE
210        CONTINUE
            DO 230 j=1,ny-9

```

```
      DO 220 i=1,nx-9
        tretz(i,j,1)=0.0
        tretz(i,j,2)=0.0
220    CONTINUE
230    CONTINUE
      RETURN
      END
```

```

c    variables.h
c234567
c    This file contains all of the variables used in the
c    subroutines comprising the main loop of the TEMAC code
c    with the exception of the far-field subroutines and the
c    ORBC subroutine. These subroutines have all their variables
c    in two other .h files (farfld.h and liao.h).
c
c*****
c
c    Variable dictionary
c
c    ddtein=constant multiplier for time derivative of incident field
c           term in lossy dielectric e-field update equations
c    ddthin=constant multiplier for time derivative of incident field
c           term in lossy magnetic h-field update equations
c    dedx=constant multiplier for lossy dielectric h-field update
c           equations
c    dedy=constant multiplier for lossy dielectric h-field update
c           equations
c    dedz=constant multiplier for lossy dielectric h-field update
c           equations
c    delt=time step
c    delto2=delt/2.0
c    dhdx=constant multiplier for lossy dielectric e-field update
c           equations
c    dhdy=constant multiplier for lossy dielectric e-field update
c           equations
c    dhdz=constant multiplier for lossy dielectric e-field update
c           equations
c    dtinv=1.0/delt
c    dtoedx=constant multiplier for free space h-field update equations
c    dtoedy=constant multiplier for free space h-field update equations
c    dtoedz=constant multiplier for free space h-field update equations
c    dtomdx=constant multiplier for free space e-field update equations
c    dtomdy=constant multiplier for free space e-field update equations
c    dtomdz=constant multiplier for free space e-field update equations
c    einc=constant multiplier for incident field term in lossy
c           dielectric e-field update equations
c    eold=constant multiplier for lossy dielectric e-field update
c           equations
c    eps=array containing permittivity values for the material types
c    exscat=scattered x-directed electric field
c    eyscat=scattered y-directed electric field
c    ezscat=scattered z-directed electric field
c    hinc=constant multiplier for incident field term in lossy
c           magnetic h-field update equations
c    hold=constant multiplier for lossy magnetic h-field update
c           equations
c    hxscat=scattered x-directed magnetic field
c    hyscat=scattered y-directed magnetic field
c    hzscat=scattered z-directed magnetic field
c    id=material type array used in reading the mesh file
c    id1=array for specifying the material type at the x-directed
c           electric field location
c    id2=array for specifying the material type at the y-directed
c           electric field location
c    id3=array for specifying the material type at the z-directed

```

```

c      electric field location
c      id4=array for specifying the material type at the x-directed
c      magnetic field location
c      id5=array for specifying the material type at the y-directed
c      magnetic field location
c      id6=array for specifying the material type at the z-directed
c      magnetic field location
c      msigma=array containing magnetic conductivity values for materials
c      mu=array containing permeability values for the material types
c      n=time step number
c      nx1=number of cells in the x direction
c      ny1=number of cells in the y direction
c      nz1=number of cells in the z direction
c      sigma=array containing electric conductivity values for materials
c      time=absolute time variable
c      xc=x coordinate of the problem space center
c      yc=y coordinate of the problem space center
c      zc=z coordinate of the problem space center

```

```

c*****

```

```

c      Now do variable typing

```

```

c      REAL exscat,eyscat,ezscat,hxscat,hyscat,hzscat
c      REAL delt,dtinv,time,xc,yc,zc,delta2
c      INTEGER n
c      INTEGER nx1,ny1,nz1,id
c      LOGICAL*1 id1,id2,id3,id4,id5,id6
c      LOGICAL*1 errflg
c      REAL dtoedx,dtoedy,dtoedz,dtomdx,dtomdy,dtomdz,eold,
c      $dhdxdy,dhdy,dhdz,dedx,dedy,dedz,einc,ddtein,hinc,ddthin,hold
c      REAL eps,mu,sigma,msigma
c      COMMON/ID/id(0:6)
c      COMMON/IDS/errflg
c      COMMON/IDSH/id1(0:nx,0:ny,0:nz),id2(0:nx,0:ny,0:nz),
c      $id3(0:nx,0:ny,0:nz),id4(0:nx,0:ny,0:nz),id5(0:nx,0:ny,0:nz),
c      $id6(0:nx,0:ny,0:nz)
c      COMMON/FIELDS/exscat(0:nx,0:ny,0:nz),eyscat(0:nx,0:ny,0:nz),
c      $ezscat(0:nx,0:ny,0:nz),hxscat(0:nx,0:ny,0:nz),
c      $hyscat(0:nx,0:ny,0:nz),hzscat(0:nx,0:ny,0:nz)
c      COMMON/TIMES/delt,dtinv,time,n,delta2
c      COMMON/GRID/xc,yc,zc
c      COMMON/GRID2/nx1,ny1,nz1
c      COMMON/UPDATE/dtoedx,dtoedy,dtoedz,dtomdx,dtomdy,dtomdz,
c      $eold(0:maxmat),dhdxdy(0:maxmat),dhdy(0:maxmat),dhdz(0:maxmat),
c      $dedxdy(0:maxmat),dedy(0:maxmat),dedz(0:maxmat),einc(0:maxmat),
c      $ddtein(0:maxmat),hinc(0:maxmat),ddthin(0:maxmat),hold(0:maxmat)
c      COMMON/MATER/eps(0:maxmat),mu(0:maxmat),sigma(0:maxmat),
c      $msigma(0:maxmat)

```

c234567

SUBROUTINE USRDEF

c

INCLUDE 'main.h'

c

c

c

c

c

c

c

c

c

c

c

c

c

c

c

c

c

c

c

c

c

c

c

c

c

c

c

c

c

c

c

c

c

c

c

c

c

c

c

c

c

c

c

c

c

c

c

c

c

c

c

c

c

c

c

c

c

c

c

c

c

c

This subroutine is where the user puts all his/her defined quantities such as material parameters and point and slice sensor locations and types.

Define material parameters here. As an example, for material type 2, the user must define eps(2), mu(2), sigma(2) and msigma(2).

eps=the permittivity of the material  
mu=the permeability of the material  
sigma=the electric conductivity of the material  
msigma=the magnetic conductivity of the material

eps(1)=(3.6855e4)\*eps0  
mu(1)=mu0  
sigma(1)=3.96e7  
msigma(1)=0.0

Define point sensors here. To define a point sensor, the user must define the following quantities:

ipt,jpt,kpt,ptsamp,ptinc

and these are described below. (See subroutine DFSENS for default values of the point sensor variables).

ipt=cell number location of point sensor in x direction (i)  
jpt=cell number location of point sensor in y direction (j)  
kpt=cell number location of point sensor in z direction (k)  
ptsamp=field quantity to be sampled  
ptinc=time increment for sampling field  
pton=time step to turn on point sensor  
ptoff=time step to turn off point sensor

For sampling field quantities, the values for ptsamp are as follows:

ptsamp=1 for sampling scattered Ex field  
ptsamp=2 for sampling scattered Ey field  
ptsamp=3 for sampling scattered Ez field  
ptsamp=4 for sampling scattered Hx field  
ptsamp=5 for sampling scattered Hy field  
ptsamp=6 for sampling scattered Hz field

\*\*\*\*\*These sensors have not been tested\*\*\*\*\*

ptsamp=7 for sampling x-directed conduction current  
ptsamp=8 for sampling y-directed conduction current  
ptsamp=9 for sampling z-directed conduction current  
ptsamp=10 for sampling x-directed displacement current

```

c      ptsamp=11 for sampling y-directed displacement current
c      ptsamp=12 for sampling z-directed displacement current
c
c*****
c      ptsamp=13 for sampling x-directed total current (Ix)
c      ptsamp=14 for sampling y-directed total current (Iy)
c      ptsamp=15 for sampling z-directed total current (Iz)
c      ptsamp=16 for sampling total Ex field
c      ptsamp=17 for sampling total Ey field
c      ptsamp=18 for sampling total Ez field
c      ptsamp=19 for sampling total Hx field
c      ptsamp=20 for sampling total Hy field
c      ptsamp=21 for sampling total Hz field
c      ptsamp=22 for sampling incident Ex field
c      ptsamp=23 for sampling incident Ey field
c      ptsamp=24 for sampling incident Ez field
c      ptsamp=25 for sampling incident Hx field
c      ptsamp=26 for sampling incident Hy field
c      ptsamp=27 for sampling incident Hz field
c      ptsamp=28 for sampling x-directed total scattered current (Ix)
c      ptsamp=29 for sampling y-directed total scattered current (Iy)
c      ptsamp=30 for sampling z-directed total scattered current (Iz)
c      ptsamp=31 for sampling x-directed total incident current (Ix)
c      ptsamp=32 for sampling y-directed total incident current (Iy)
c      ptsamp=33 for sampling z-directed total incident current (Iz)
c
c      Note: the variable ptinc permits you to sample a field
c            quantity at a variable time step increment. For example,
c            if you only need to sample a particular field quantity every
c            10 time steps, then set ptinc=10. For sampling every time
c            step, ptinc=1 (this is the default).
c
c      The variables pton and ptoff tell at what time step to turn
c            on and off the point source sampling.
c
c      After defining the above variables, to set the point sensor,
c            issue a call to subroutine PTSNSR and pass the sensor number
c            as the argument to the subroutine. An example is shown below:
c
c      ipt=10
c      jpt=10
c      kpt=10
c      ptsamp=13
c      ptinc=1
c      pton=1
c      ptoff=tsteps
c      CALL PTSNSR(1)
c
c      ipt=25
c      jpt=25
c      kpt=25
c      ptsamp=1
c      ptinc=1
c      pton=1
c      ptoff=tsteps
c      CALL PTSNSR(1)

```



```

c      slplan=3:
c
c      ^
c      |
c      |
c      |
c      |-----> y (imin,imax)
c      |
c      /
c     x

```

In order to define slsamp correctly, please see the above comments regarding ptsamp. The correct values for slsamp will be the same as those for ptsamp.

The variable slinc permits you to save the slice data only once or at particular time step increments.

slinc=0 to save the slice data only once  
 slinc=10 to save the slice data every 10 time steps

The variables slon and sloff define the time step to turn the slice saving on and/or off (sloff is ignored for saving only once, but slon MUST be defined).

An example definition is given below:

This slice sensors is defined as an xy plane located at  $nz/2+1$ , to sample over the entire plane ( $i=1$  to  $nx$  and  $j=1$  to  $ny$ ), and to sample x-directed electric field at time step 100.

```

c      slplan=1
c      slloc=nz/2+1
c      imin=1
c      imax=ny
c      jmin=1
c      jmax=nx
c      slsamp=1
c      slinc=0
c      slon=100
c      sloff=tsteps
c      CALL SLSNSR(1)

c
c      slplan=1
c      slloc=nz/2+1
c      imin=1
c      imax=nx
c      jmin=25
c      jmax=25
c      slsamp=2
c      slinc=0
c      slon=750
c      sloff=750
c      CALL SLSNSR(1)

c      RETURN
c      END

```

```

c234567
c
      SUBROUTINE STLIAO
c
      INCLUDE 'main.h'
c
      INTEGER fact(0:order),sign
      INTEGER i,j,l,ioff
      REAL*8 binom(0:order),sx,sy,sz,tx(0:2*order+1,0:order),
      $ty(0:2*order+1,0:order),tz(0:2*order+1,0:order)
c
      This subroutine initializes the Liao ORBC
c
c*****
c
      Local variable dictionary
c
      binom=binomial coefficient used in Liao ORBC
      fact=array for computing factorials
      i=loop counter over spatial indices
      ioff=offset variable used to determine l
      j=loop counter over order number
      l=index variable into interpolation matrix T
      m=loop counter over time steps for filling the
      back time step array
      sign=variable used to evaluate a  $(-1)^{(n+1)}$  type
      of term
      sx=Courant number based upon delx
      sy=Courant number based upon dely
      sz=Courant number based upon delz
      tx=interpolation matrix T based upon sx
      ty=interpolation matrix T based upon sy
      tz=interpolation matrix T based upon sz
c
c*****
c
      Zero the interpolation matrix and initialize
      the factorial array.
c
      fact(0)=1
      DO 30 j=1,order
        fact(j)=fact(j-1)*j
        DO 20 i=1,2*j+1
          tx(i,j)=0.0
          ty(i,j)=0.0
          tz(i,j)=0.0
          bcx(i,j)=0.0
          bcy(i,j)=0.0
                                     bcz(i,j)=0.0
        20   CONTINUE
      30   CONTINUE
c
      Compute the Courant numbers
c
      sx=c*delt/delx
      sy=c*delt/dely
      sz=c*delt/delz
c

```

```

c      Fill in the (1,1), (2,1) and (3,1) interpolation
c      matrix terms
c
      tx(1,1)=0.5D00*(2.0D00*0.9925-sx)*(1.0-sx)
      tx(2,1)=sx*(2.0D00-sx)
      tx(3,1)=0.5D00*sx*(sx-1.0D00)
      ty(1,1)=0.5D00*(2.0D00*0.9925-sy)*(1.0D00-sy)
      ty(2,1)=sy*(2.0D00-sy)
      ty(3,1)=0.5D00*sy*(sy-1.0D00)
      tz(1,1)=0.5*(2.0D00*0.9925-sz)*(1.0D00-sz)
      tz(2,1)=sz*(2.0D00-sz)
      tz(3,1)=0.5D00*sz*(sz-1.0D00)
c
c      Finish filling in the interpolation matrix
c
      ioff=2*order-2
      DO 60 j=2,order
        DO 50 i=1,2*j+1
          DO 40 l=max(1,i-ioff),min(i,3)
            tx(i,j)=tx(i,j)+tx(i-l+1,j-l)*tx(l,1)
            ty(i,j)=ty(i,j)+ty(i-l+1,j-l)*ty(l,1)
            tz(i,j)=tz(i,j)+tz(i-l+1,j-l)*tz(l,1)
40          CONTINUE
          50      CONTINUE
60      CONTINUE
c
c      Compute the binomial and sign terms
c
      DO 80 j=1,order
        binom(j)=REAL(fact(order)/(fact(order-j)*fact(j)))
        sign=(-1)**(j+1)
        DO 70 i=1,2*j+1
c
c          Now compute the constant multiplier terms
c
          bcx(i,j)=REAL(sign)*binom(j)*tx(i,j)
          bcy(i,j)=REAL(sign)*binom(j)*ty(i,j)
          bcz(i,j)=REAL(sign)*binom(j)*tz(i,j)
70      CONTINUE
80      CONTINUE
c
c      Close the diagnostics file
c
      CLOSE (UNIT=10)
      RETURN
      END

```

c234567

```
c      This file contains all of the source functions for the
c      source pulse type.
c
c      IF (gauss) THEN
c
c          Gaussian pulse type
c
c          source=exp(-((tprime-toff)*tau0i)**2)
c      ELSEIF (banlim) THEN
c
c          Bandlimited pulse type
c
c          source=exp(-((tprime-toff)*tau0i)**2)*sin(w0*(tprime-toff))
c          source=cos(w0*(tprime-toff))*
+          sin(wup*(tprime-toff))/(wup*(tprime-toff))
c      ELSEIF (hypsec) THEN
c
c          Hyperbolic secant pulse type
c
c          source=1.0/(0.5*(exp(-(tprime-toff)*tau0i)+
c          $exp((tprime-toff)*tau0i)))
c      ELSEIF (rsine) THEN
c
c          Ramped sinusoid
c
c          source=(1.0-exp(-((tprime-toff)*tau0i)**2))*sin(w0*(tprime-
c          $toff))
c      ELSEIF (step) THEN
c
c          Unit step
c
c          source=1.0-exp(-((tprime-toff)*tau0i)**2)
c      ENDIF
```

```

c      setup.h
c234567
c      This file contains all of the parameters that the
c      user has control over when running different problems.
c      This file along with the 'userdefs.f' and 'geom.f' files
c      will be the only files requiring modification to specify a
c      new problem.
c
c
c*****
c
c      Variable definitions (alphabetical order)
c
c      ampl=the amplitude of the point source or plane wave
c             forcing function.
c      banlim=flag used to select the bandlimited time domain
c             function
c      delx=the cell size in the x direction (in meters)
c      dely=the cell size in the y direction (in meters).
c      delz=the cell size in the z direction (in meters).
c      ffldon=flag used to turn on/off far field transformation
c      freesp=flag used to run only free space within the problem
c             space
c      fspec=flag used for specifying incident excitation in freq.
c      gauss=flag used to select the Gaussian time domain function
c      hypsec=flag used to select the hyperbolic secant time domain
c             function
c      maxmat=the maximum number of materials to be defined
c      mmax=the maximum number of time bins for the far-field
c            vector potential arrays
c      nrfld=flag used to turn on/off near field data sampling
c            domain
c      nx=the number of CELLS in the x direction+1.
c      ny=the number of CELLS in the y direction+1.
c      nz=the number of CELLS in the z direction+1.
c      order=order of Liao Outer Radiation Boundary Condition
c      phi=far field observation point phi angle
c      phiin=incidence angle in the phi-direction for an incident
c             plane wave source (in degrees measured from the x-axis).
c             See note.
c      plwave=flag used for turning on/off the incident plane wave
c      pntsrc=flag used for turning on point source
c      point=flag used to turn on/off point sensor sampling
c      rdmesh=flag used to turn on/off reading of geometry mesh files
c      rise=flag used for specifying source pulse by rise time
c      rsine=flag used to select the ramped sinusoid time domain
c             function
c      step=unit step function
c      theta=far field observation point theta angle
c      thin=incidence angle in the theta-direction for an
c            incident plane wave source (in degrees measured from the
c            z-axis). See note.
c      thpol=amount of polarization in the theta-direction.
c      tspec=flag used for specifying incident excitation in time
c            domain
c      tsteps=total number of time steps desired for execution.
c
c      Note: plane waves are propagated at the phiin and thin angles

```

```

C          towards the origin
C*****
C
C          Variable type specifications
C
C              INTEGER nx,ny,nz
C              INTEGER tsteps,mmax
C              REAL delx,dely,delz,thin,phiin,thpol,ampl
C              LOGICAL*1 plwave,tspec,fspec,gauss,rise,banlim,rsine,hypsec,
C              $step
C              LOGICAL*1 rdmesh,ffldon,nrfld,freeesp,pntsrc
C              INTEGER npoint,nslice
C              INTEGER order,maxmat
C              REAL theta,phi
C
C          PARAMETER (nx=50,ny=50,nz=50)
C
C          Reset tsteps after first run
C
C          PARAMETER (tsteps=1000)
C          PARAMETER (delx=0.01,dely=0.01,delz=0.01)
C          PARAMETER (thin=22.5,phiin=22.5)
C          PARAMETER (thpol=0.0,ampl=1.0)
C
C          Rise = false could cause problem
C
C          PARAMETER (plwave=true,pntsrc=1-plwave,tspec=false,
C          $fspec=1-tspec,rsine=false,gauss=true,banlim=false,hypsec=false,
C          $rsine=false,step=false)
C          PARAMETER (rdmesh=false,nrfld=false,ffldon=true,freeesp=false)
C          PARAMETER (order=2)
C          PARAMETER (npoint=1,nslice=1)
C          PARAMETER (maxmat=2)
C          PARAMETER (theta=0.0,phi=0.0,mmax=10000000)

```

c234567

```
c
c      SUBROUTINE SETUP
c
c      INCLUDE 'main.h'
c      INTEGER m,is
c      CHARACTER*55 typtxt
c
c      This subroutine initializes update equation multipliers,
c      material constitutive parameters and other necessary
c      variables and writes a diagnostics file.
c
c      *****
c
c      Local variable dictionary
c
c      is=loop counter for writing out point and/or slice sensor
c      information
c      m=loop counter over material types for setting constant
c      multipliers for various materials
c      typtxt=text string to indicate the sensor type for the
c      point and/or slice sensors
c
c      *****
c
c      Open the diagnostics file
c
c      OPEN (UNIT=10,FILE='info3d.dat',STATUS='UNKNOWN')
c
c      Compute the center coordinates
c
c      xc=0.5*nx1*delx
c      yc=0.5*ny1*dely
c      zc=0.5*nz1*delz
c
c      Compute the time step and its inverse.
c
c      delt=1.0/(c*SQRT(1.0/(delx*delx)+1.0/(dely*dely)+
c      $1.0/(delz*delz)))
c      delt=0.01*delt
c      delto2=0.5*delt
c      dtinv=1.0/delt
c
c      Compute cosine and sines of incidence angles
c
c      IF ((abs(thin).EQ.90.).OR.(abs(thin).EQ.270.)) THEN
c          costh=0.0
c      ELSEIF(thin.EQ.0.) THEN
c          costh=1.0
c      ELSEIF(abs(thin).EQ.180.) THEN
c          costh=-1.0
c      ELSE
c          costh=COS(degrad*thin)
c      ENDIF
c      IF ((abs(thin).EQ.0.).OR.(abs(thin).EQ.180.)) THEN
c          sinth=0.0
c      ELSEIF((thin.EQ.90.).OR.(thin.EQ.-270.)) THEN
c          sinth=1.0
```

```

ELSEIF((thin.EQ.270.).OR.(thin.EQ.-90.)) THEN
    sinh=-1.0
ELSE
    sinh=SIN(degrad*thin)
ENDIF
IF ((abs(phiin).EQ.90.).OR.(abs(phiin).EQ.270.)) THEN
    cosphi=0.0
ELSEIF(phiin.EQ.0.) THEN
    cosphi=1.0
ELSEIF(abs(phiin).EQ.180.) THEN
    cosphi=-1.0
ELSE
    cosphi=COS(degrad*phiin)
ENDIF
IF ((abs(phiin).EQ.0.).OR.(abs(phiin).EQ.180.)) THEN
    sinphi=0.0
ELSEIF((phiin.EQ.90.).OR.(phiin.EQ.-270.)) THEN
    sinphi=1.0
ELSEIF((phiin.EQ.270.).OR.(phiin.EQ.-90.)) THEN
    sinphi=-1.0
ELSE
    sinphi=SIN(degrad*phiin)
ENDIF
C
C   Compute amount of phi-polarization based upon theta-
C   polarization
C
    phipol=SQRT(1.0-thpol*thpol)
C
C   Compute amplitude of incident plane wave vector in
C   each direction
C
    eamplx=ampl*(thpol*costh*cosphi-hipol*sinphi)
    eamply=ampl*(thpol*costh*sinphi+hipol*cosphi)
    eamplz=ampl*(-thpol*sinth)
    hamplx=ampl/eta0*(hipol*costh*cosphi+thpol*sinphi)
    hamply=ampl/eta0*(hipol*costh*sinphi-thpol*cosphi)
    hamplz=ampl/eta0*(-hipol*sinth)
    samplx=eamply*hamplz-eamplz*hamply
    samply=eamplz*hamplx-eamplx*hamplz
    samplz=eamplx*hamply-eamply*hamplx
C
C   Compute direction cosines of incident wave vector
C
    cosa=-cosphi*sinth
    cosb=-sinphi*sinth
    cosg=-costh
    dxcosa=delx*cosa
    dycosb=dely*cosb
    dzcosg=delz*cosg
C
C   Compute time offset for incident plane wave propagation
C   into computational space.
C
    tdelay=0.0
    IF (cosa.LT.0.) tdelay=tdelay-cosa*nxl*delx*cinv
    IF (cosb.LT.0.) tdelay=tdelay-cosb*nyl*dely*cinv
    IF (cosg.LT.0.) tdelay=tdelay-cosg*nzl*delz*cinv

```

```

C      Set up the defaults, source pulse information and
C      user definitions
C
      CALL DEFLT5
      write (*,*) 'finished defaults'
      CALL USRDEF
      write (*,*) 'finished userdefs'
      CALL PULSE
      write (*,*) 'finished pulse'

C      Initialize update equation multipliers
C
C      Free space multipliers first
C
      dtoedx=delt/(eps0*delx)
      dtoedy=delt/(eps0*dely)
      dtoedz=delt/(eps0*delz)
      dtomdx=delt/(mu0*delx)
      dtomdy=delt/(mu0*dely)
      dtomdz=delt/(mu0*delz)

C      Now lossy dielectric and lossy magnetic multipliers
C
      DO 10 m=2,maxmat
        eold(m)=eps(m)/(eps(m)+sigma(m)*delt)
        einc(m)=sigma(m)*delt/(eps(m)+sigma(m)*delt)
        ddtein(m)=delt*(eps(m)-eps0)/(eps(m)+sigma(m)*delt)
        dhdx(m)=delt/((eps(m)+sigma(m)*delt)*delx)
        dhdy(m)=delt/((eps(m)+sigma(m)*delt)*dely)
        dhdz(m)=delt/((eps(m)+sigma(m)*delt)*delz)
        hold(m)=mu(m)/(mu(m)+msigma(m)*delt)
        hinc(m)=msigma(m)*delt/(mu(m)+msigma(m)*delt)
        ddthin(m)=delt*(mu(m)-mu0)/(mu(m)+msigma(m)*delt)
        dedx(m)=delt/((mu(m)+msigma(m)*delt)*delx)
        dedy(m)=delt/((mu(m)+msigma(m)*delt)*dely)
        dedz(m)=delt/((mu(m)+msigma(m)*delt)*delz)
10    CONTINUE

C      Write information file now
C
c23456789012345678901234567890123456789012345678901234567890123456789012
C
C      Write grid information first
C
      WRITE (10,*) '*****'
      WRITE (10,*) '                                Grid information'
      WRITE (10,*) ' '
      WRITE (10,*) '                                Problem space size'
      WRITE (10,*) ' '
      WRITE (10,*) 'Nx =',nx1,' cells in the x direction.'
      WRITE (10,*) 'Ny =',ny1,' cells in the y direction.'
      WRITE (10,*) 'Nz =',nz1,' cells in the z direction.'
      WRITE (10,*) ' '
      WRITE (10,*) '                                Cell size'
      WRITE (10,*) ' '
      WRITE (10,*) 'Delx =',delx,' meters in the x direction.'
      WRITE (10,*) 'Dely =',dely,' meters in the y direction.'

```

```

WRITE (10,*) 'Delz =',delz,' meters in the z direction.'
WRITE (10,*) ' '
WRITE (10,*) ' ' Time step'
WRITE (10,*) ' '
WRITE (10,*) 'Delt =',delt*1.0e12,' picoseconds.'
WRITE (10,*) '*****'

```

C  
C  
C

Write plane wave information if plane wave is selected

```

IF (plwave) THEN
    WRITE (10,*) ' ' Incident plane wave information'
    WRITE (10,*) ' '
    WRITE (10,*) 'Amount of theta polarization =',thpol
    WRITE (10,*) 'Amount of phi polarization =',phipol
    WRITE (10,*) ' '
    WRITE (10,*) 'Theta incidence angle =',thin,' degrees.'
    WRITE (10,*) 'Phi incidence angle =',phiin,' degrees.'
    WRITE (10,*) ' '
    WRITE (10,*) 'Amplitude of incident plane wave =',ampl
    WRITE (10,*) 'Amplitude of incident Ex component =',eamplx,
$' V/m.'
    WRITE (10,*) 'Amplitude of incident Ey component =',eamply,
$' V/m.'
    WRITE (10,*) 'Amplitude of incident Ez component =',eamplz,
$' V/m.'
    WRITE (10,*) 'Amplitude of incident Hx component =',hamplx,
$' A/m.'
    WRITE (10,*) 'Amplitude of incident Hy component =',hamply,
$' A/m.'
    WRITE (10,*) 'Amplitude of incident Hz component =',hamplz,
$' A/m.'
    WRITE (10,*) ' '
    WRITE (10,*) 'Poynting Vector:'
    WRITE (10,*) ' '
    WRITE (10,*) 'Amplitude of incident Sx component =',samplx,
$' W/m**2.'
    WRITE (10,*) 'Amplitude of incident Sy component =',samply,
$' W/m**2.'
    WRITE (10,*) 'Amplitude of incident Sz component =',samlz,
$' W/m**2.'
    WRITE (10,*) ' '
    WRITE (10,*) 'Time delay =',tdelay*1.0e12,' picoseconds.'
    WRITE (10,*) ' '

```

ELSE

C  
C  
C  
C

Indicate plane wave is turned off and write  
point source information

```

WRITE (10,*) ' '
WRITE (10,*) 'Plane wave is turned off.'
WRITE (10,*) ' '
WRITE (10,*) 'Point source information:'
WRITE (10,*) ' '
WRITE (10,*) 'Feed type is ',fdtype,' directed electric',
$' field.'
WRITE (10,*) 'i location =',iptsrc
WRITE (10,*) 'j location =',jptsrc
WRITE (10,*) 'k location =',kptsrc

```

```

WRITE (10,*) ' '
ENDIF

C
C Now write information pertaining to the source type
C
WRITE (10,*) '*****'
WRITE (10,*) 'Source information'
WRITE (10,*) ' '
IF (gauss) THEN
WRITE (10,*) 'Pulse type is Gaussian.'
ELSEIF (banlim) THEN
WRITE (10,*) 'Pulse type is bandlimited pulse.'
ELSEIF (hypsec) THEN
WRITE (10,*) 'Pulse type is Hyperbolic secant.'
ELSEIF (rsine) THEN
WRITE (10,*) 'Pulse type is ramped sine.'
ELSEIF (step) THEN
WRITE (10,*) 'Pulse type is unit step.'
ELSE
WRITE (10,*) 'Error! No incident plane wave time domain'
WRITE (10,*) 'function is specified. Please check the '
WRITE (10,*) '"setup.h" file. Execution halted.'
STOP
ENDIF
IF (rsine) THEN

C
C Ramped sinusoid source information
C
WRITE (10,*) 'Rise time =',trise,' seconds.'
WRITE (10,*) 'Rise time =',nrise,' time steps.'
WRITE (10,*) 'Frequency =',f0,' Hz.'
WRITE (10,*) 'Period =',1.0/f0,' seconds.'
WRITE (10,*) 'Time steps per period =',nper
GO TO 100
ELSEIF (step) THEN

C
C Pulse information
C
WRITE (10,*) 'Rise time =',trise,' seconds.'
WRITE (10,*) 'Rise time =',nrise,' time steps.'
WRITE (10,*) 'Half-power pulse width =',thp,' seconds.'
WRITE (10,*) 'Half-power pulse width =',nhp,' time steps.'
WRITE (10,*) 'HWHM pulse width =',tfwhm,' seconds.'
WRITE (10,*) 'HWHM pulse width =',nfwhm,' time steps.'
WRITE (10,*) '1/e pulse width =',te,' seconds.'
WRITE (10,*) '1/e pulse width =',nte,' time steps.'
ELSE

C
C Pulse information
C
WRITE (10,*) 'Rise time =',trise,' seconds.'
WRITE (10,*) 'Rise time =',nrise,' time steps.'
WRITE (10,*) 'Half-power pulse width =',thp,' seconds.'
WRITE (10,*) 'Half-power pulse width =',nhp,' time steps.'
WRITE (10,*) 'FWHM pulse width =',tfwhm,' seconds.'
WRITE (10,*) 'FWHM pulse width =',nfwhm,' time steps.'
WRITE (10,*) '1/e pulse width =',te,' seconds.'
WRITE (10,*) '1/e pulse width =',nte,' time steps.'

```

```

WRITE (10,*) 'Truncation pulse width =',ttrun,' seconds.'
WRITE (10,*) 'Truncation pulse width =',ntrun,' time steps.'
WRITE (10,*) 'Time offset =',toff,' seconds.'
WRITE (10,*) 'Time offset =',ntoff,' time steps.'
IF (gauss) THEN
  IF (tspec) THEN
    WRITE (10,*) '80 dB frequency limit =',f80,' Hz'
  ELSE
    WRITE (10,*) 'Attenuation at upper frequency limit =',
$gdb,' dB.'
  ENDIF
  WRITE (10,*) 'Upper frequency limit =',fup,' Hz.'
ELSEIF (banlim) THEN
  IF (tspec) THEN
    WRITE (10,*) '80 dB frequency limit =',f80,' Hz'
  ELSE
    WRITE (10,*) 'Attenuation at upper frequency limit =',
$bldb,' dB.'
  ENDIF
  WRITE (10,*) 'Upper frequency limit =',fup,' Hz.'
ENDIF
ENDIF
C
C   Now write point and/or slice sensor information
C
100 WRITE (10,*) '*****'
WRITE (10,*) 'Sensor information'
WRITE (10,*) ' '
C
C   Point sensor information
C
WRITE (10,*) 'Number of point sensors selected = ',npoint
DO 20 is=1,npoint
  WRITE (10,*) ' '
  WRITE (10,*) 'Point sensor # ',is
  WRITE (10,*) 'i location = ',ptsens(is,1)
  WRITE (10,*) 'j location = ',ptsens(is,2)
  WRITE (10,*) 'k location = ',ptsens(is,3)
  WRITE (10,*) 'Sensor type = ',ptsens(is,4)
  CALL SAMTYP(ptsens(is,4),typtxt)
  WRITE (10,*) typtxt
  WRITE (10,*) 'Time step saving increment = ',ptsens(is,5)
  WRITE (10,*) 'Turn on time step = ',ptsens(is,6)
  WRITE (10,*) 'Turn off time step = ',ptsens(is,7)
20 CONTINUE
C
C   Slice sensor information
C
WRITE (10,*) ' '
WRITE (10,*) 'Number of slice sensors selected = ',nslice
DO 30 is=1,nslice
  slplan=slsens(is,1)
  WRITE (10,*) ' '
  WRITE (10,*) 'Slice sensor # ',is
  WRITE (10,*) 'Slice plane type = ',slplan
  IF (slplan.EQ.1) THEN
    WRITE (10,*) 'Slice plane is parallel to xy plane.'
    WRITE (10,*) 'Slice plane is located at k = ',slsens(is,2)
  
```

```

        WRITE (10,*) 'Slice plane samples from i = ',slsens(is,3),
$' to i = ',slsens(is,4)
        WRITE (10,*) 'and from j = ',slsens(is,5),' to j = ',
$slsens(is,6)
        ELSEIF (slplan.EQ.2) THEN
            WRITE (10,*) 'Slice plane is parallel to xz plane.'
            WRITE (10,*) 'Slice plane is located at j = ',slsens(is,2)
            WRITE (10,*) 'Slice plane samples from k = ',slsens(is,3),
$' to k = ',slsens(is,4)
            WRITE (10,*) 'and from i = ',slsens(is,5),' to i = ',
$slsens(is,6)
        ELSEIF (slplan.EQ.3) THEN
            WRITE (10,*) 'Slice plane is parallel to yz plane.'
            WRITE (10,*) 'Slice plane is located at i = ',slsens(is,2)
            WRITE (10,*) 'Slice plane samples from j = ',slsens(is,3),
$' to j = ',slsens(is,4)
            WRITE (10,*) 'and from k = ',slsens(is,5),' to k = ',
$slsens(is,6)
        ENDIF
        WRITE (10,*) 'Sensor type = ',slsens(is,7)
        CALL SAMTYP(slsens(is,7),typtxt)
        WRITE (10,*) typtxt
        IF (slsens(is,8).EQ.0) THEN
            WRITE (10,*) 'Slice sensor saved only once at time step ',
$slsens(is,9)
        ELSE
            WRITE (10,*) 'Time step saving increment = ',slsens(is,8)
            WRITE (10,*) 'Turn on time step = ',slsens(is,9)
            WRITE (10,*) 'Turn off time step = ',slsens(is,10)
        ENDIF
30    CONTINUE
        WRITE (10,*) '*****'
c
c    Write source function and time derivative source function
c    to data files
c
        OPEN (UNIT=35,FILE='source.dat',STATUS='UNKNOWN')
        OPEN (UNIT=36,FILE='ddtsrce.dat',STATUS='UNKNOWN')
        WRITE (35,1000)
        WRITE (35,1001)
        WRITE (35,1002)
        WRITE (35,1003)
        WRITE (35,1004)
        WRITE (36,1005)
        WRITE (36,1006)
        WRITE (36,1007)
        WRITE (36,1008)
        WRITE (36,1009)
        WRITE (36,1010)
        DO 200 n=1,tsteps
            time=(n-1)*delt
            tprime=time
            INCLUDE 'source.h'
            INCLUDE 'ddtsrce.h'
            WRITE (35,*) time*1.0E9,n,source
            WRITE (36,*) time*1.0E9,n,ddtsrc
200    CONTINUE
        time=0.0

```

```

        CLOSE (UNIT=35)
        CLOSE (UNIT=36)
1000 FORMAT(T1,'# This file contains the source function value')
1001 FORMAT(T1,'# versus time and time steps. The format of')
1002 FORMAT(T1,'# the data file is as follows:')
1003 FORMAT(T1,'# Time (ns) Time step Source function value')
1004 FORMAT(T1,'#-----')
1005 FORMAT(T1,'# This file contains the time derivative of')
1006 FORMAT(T1,'# the source function value versus time and')
1007 FORMAT(T1,'# time steps. The format of the data file is')
1008 FORMAT(T1,'# as follows:')
1009 FORMAT(T1,'# Time (ns) Time step d/dt(Source function',
        $' value')
1010 FORMAT(T1,'#-----')
        RETURN
        END

```

c234567

C

SUBROUTINE SETCON

C

INCLUDE 'constants.h'

C

C

C

This subroutine sets all the necessary constants  
used throughout the TEMAC code.

C

pi=4.0\*ATAN(1.0)  
twopi=2.0\*pi  
eps0=1.0/(36.0\*pi)\*1.0e-9  
mu0=4.0\*pi\*1.0e-7  
eta0=SQRT(mu0/eps0)  
c=1.0/SQRT(eps0\*mu0)  
cinv=1.0/c  
degrad=pi/180.0  
e=EXP(1.0)  
RETURN  
END

```

C      SUBROUTINE STCELL(istart,jstart,kstart,nxwide,nywide,nzwide,mtype)
C
C      INCLUDE 'main.h'
C      INTEGER istart,jstart,kstart,nxwide,nywide,nzwide,iend,
C      $jend,kend,i,j,k
C      LOGICAL*1 mtype
C
C      This subroutine sets twelve id? components for one Yee cell
C      to the same material type specified by mtype.  If nxwide, nywide,
C      or nzwide=0, then only 4 id? components will be set corresponding
C      to a sheet of two-dimensional Yee cells with zero thickness.
C
C*****
C
C      Local variable dictionary
C
C      i=cell coordinate number in x direction
C      iend=ending cell coordinate number in x direction
C      istart=starting cell coordinate number in x direction
C      j=cell coordinate number in y direction
C      jend=ending cell coordinate number in y direction
C      jstart=starting cell coordinate number in y direction
C      k=cell coordinate number in z direction
C      kend=ending cell coordinate number in z direction
C      kstart=starting cell coordinate number in z direction
C      mtype=material type number
C      nxwide=number of cells in x direction to set
C      nywide=number of cells in y direction to set
C      nzwide=number of cells in z direction to set
C
C*****
C
C      iend=istart+nxwide-1
C      jend=jstart+nywide-1
C      kend=kstart+nzwide-1
C
C      IF (nxwide.EQ.0) THEN
C
C          Set a sheet of zero thickness parallel to yz plane
C
C          DO 20 k=kstart,kend
C              DO 10 j=jstart,jend
C                  id2(istart,j,k)=mtype
C                  id2(istart,j,k+1)=mtype
C                  id3(istart,j,k)=mtype
C                  id3(istart,j+1,k)=mtype
C              CONTINUE
C          CONTINUE
C          ELSEIF (nywide.EQ.0) THEN
C
C          Set a sheet of zero thickness parallel to xz plane
C
C          DO 40 k=kstart,kend
C              DO 30 i=istart,iend
C                  id1(i,jstart,k)=mtype
C                  id1(i,jstart,k+1)=mtype
C                  id3(i,jstart,k)=mtype

```

```

        id3(i+1,jstart,k)=mtype
30      CONTINUE
40      CONTINUE
      ELSEIF (nzwide.EQ.0) THEN
c
c      Set a sheet of zero thickness parallel to xy plane
c
      DO 60 j=jstart,jend
        DO 50 i=istart,iend
          id1(i,j,kstart)=mtype
          id1(i,j+1,kstart)=mtype
          id2(i,j,kstart)=mtype
          id2(i+1,j,kstart)=mtype
50      CONTINUE
60      CONTINUE
      ELSE
c
c      Set a block of Yee cells (12 components/cell)
c
      DO 90 k=kstart,kend
        DO 80 j=jstart,jend
          DO 70 i=istart,iend
            id1(i,j,k)=mtype
            id1(i,j,k+1)=mtype
            id1(i,j+1,k+1)=mtype
            id1(i,j+1,k)=mtype
            id2(i,j,k)=mtype
            id2(i+1,j,k)=mtype
            id2(i+1,j,k+1)=mtype
            id2(i,j,k+1)=mtype
            id3(i,j,k)=mtype
            id3(i+1,j,k)=mtype
            id3(i+1,j+1,k)=mtype
            id3(i,j+1,k)=mtype
70      CONTINUE
80      CONTINUE
90      CONTINUE
      ENDIF
      RETURN
      END

```

```

c234567
c      Include file sensors.h
c      This file contains all of the variables used for defining
c      point or slice sensors.
c
c*****
c
c      Variable dictionary
c
c      imin=minimum cell number for saving slice sensor information
c           in the "x" direction
c      imax=maximum cell number for saving slice sensor information
c           in the "x" direction
c      ipt=used for defining i location of point or slice sensor
c      jmin=minimum cell number for saving slice sensor information
c           in the "y" direction
c      jmax=maximum cell number for saving slice sensor information
c           in the "y" direction
c      jpt=used for defining j location of point or slice sensor
c      kpt=used for defining k location of point or slice sensor
c      ptinc=time step increment for saving point sensor data
c      ptsamp=used for defining point sensor sample type
c      ptsens=array used to store information about a point sensor
c      slinc=time step increment for saving slice sensor data
c      slloc=location of slice sensor in third dimension
c      slplan=used to define the slice sensor plane (1,2 or 3)
c      slsamp=used for defining slice sensor sample type
c      slsens=array used to store information about a slice sensor
c*****
c
c      INTEGER ptsens,slsens,pton,ptoff,slon,sloff
c      INTEGER ptsens,slsens,ipt,jpt,kpt,ptsamp,ptinc,
c      $slplan,slloc,imin,imax,jmin,jmax,slsamp,slinc,pton,ptoff,
c      $slon,sloff
c
c      COMMON/SENSR/ptsens(0:npoint,0:7),slsens(0:nslice,0:10),
c      $pton,ptoff,slon,sloff,ipt,jpt,kpt,ptsamp,ptinc,slplan,slloc,
c      $imin,imax,jmin,jmax,slsamp,slinc
c      COMMON/SENSR2/ipt,jpt,kpt,ptsamp,ptinc,slplan,slloc,
c      $imin,imax,jmin,jmax,slsamp,slinc

```

```

c234567
      SUBROUTINE DFSENS
C
      INCLUDE 'main.h'
      INTEGER is
C
C      This subroutine sets the defaults for the point and/or
C      slice sensors.
C
C*****
C
C      Local variable dictionary
C
C      is=loop counter for defining point or slice sensors
C
C*****
C
C      First set the defaults for the point sensors. All sensors
C      are defaulted to the center of the problem space and will
C      sample the x-directed electric field at every time step.
C
      DO 10 is=1,npoint
          ipt=nx/2+1
          jpt=ny/2+1
          kpt=nz/2+1
          ptsamp=1
          ptinc=1
          pton=1
          ptoff=tsteps
          CALL PTSNSR(is)
10     CONTINUE
C
C      Now set the defaults for the slice sensors. All slice
C      sensors are defaulted to an xy plane located at nz/2+1,
C      sample over the entire plane (i=1 to nx and j=1 to ny),
C      and will sample x-directed electric field at time step 100.
C
      DO 20 is=1,nslice
          slplan=1
          slloc=nz/2+1
          imin=1
          imax=nx
          jmin=1
          jmax=ny
          slsamp=1
          slinc=0
          slon=100
          sloff=tsteps
          CALL SLSNSR(is)
20     CONTINUE
      RETURN
      END
C
      SUBROUTINE PTSNSR(is)
C
      INCLUDE 'main.h'
      INTEGER is
C

```

```

c      This subroutine sets the point sensor
c      array values corresponding to what the
c      user has defined.
c
c*****
c
c      Local variable dictionary
c
c      is=sensor number
c
c*****
c
c      if(is .gt. npoint)then
c          write(*,*)'                ERROR !!!!!!!!'
c          write(*,*)
c          write(*,*)'Point sensor call',is,'from userdefs.f is greater'
c          write(*,*)'than npoint in setup.h. Terminationg code run'
c          stop
c      endif
c      ptsens(is,1)=ipt
c      ptsens(is,2)=jpt
c      ptsens(is,3)=kpt
c      ptsens(is,4)=ptsamp
c      ptsens(is,5)=ptinc
c      ptsens(is,6)=pton
c      ptsens(is,7)=ptoff
c      IF(ptinc.EQ.0)ptsens(is,7)=pton
c      RETURN
c      END
c
c      SUBROUTINE SLSNSR(is)
c
c      INCLUDE 'main.h'
c      INTEGER is
c
c      This subroutine sets the slice sensor
c      array values corresponding to what the
c      user has defined.
c
c*****
c
c      Local variable dictionary
c
c      is=sensor number
c
c*****
c
c      if(is .gt. nslice)then
c          write(*,*)'                ERROR !!!!!!!!'
c          write(*,*)
c          write(*,*)'Slice sensor call',is,'from userdefs.f is greater'
c          write(*,*)'than nslice in setup.h. Terminationg code run'
c          stop
c      endif
c      slsens(is,1)=slplan
c      slsens(is,2)=slloc
c      slsens(is,3)=imin
c      slsens(is,4)=imax

```

```

      slsens(is,5)=jmin
      slsens(is,6)=jmax
      slsens(is,7)=slsamp
      slsens(is,8)=slinc
      slsens(is,9)=slon
      slsens(is,10)=sloff
      RETURN
      END

C
      SUBROUTINE SENSOR
C
      INCLUDE 'main.h'
      INTEGER*2 is
      INTEGER i,j,k
      INTEGER num
      CHARACTER*14 ptfile,slfile
      REAL field

C
      This subroutine samples and saves point and/or
      slice sensor data
C
C*****
C
      Local variable dictionary
C
      field=field quantity that has been sampled
      i=cell coordinate number in x direction
      is=loop counter over the number of point and/or slice
      sensors
      j=cell coordinate number in y direction
      k=cell coordinate number in z direction
      ptfile=file name for point sensor data
      slfile=file name for slice sensor data
C
C*****
C
      Save point sensors
C
      DO 10 is=1,npoint

C
      Open a data file for point sensor and save the
      field data
C
      Do the file name first
C
      IF (n.GE.ptsens(is,6).AND.n.LE.ptsens(is,7).AND.
+      (MOD((n-ptsens(is,6)),ptsens(is,5)).EQ.0)) THEN
        ptfile(1:6)='ptsens'
        IF (is.lt.10) THEN
          ptfile(7:7)='0'
          write(ptfile(8:8),'(i1)')is
        ELSE
          write(ptfile(7:8),'(i2)')is
        ENDIF
        ptfile(9:12)='.dat'
        OPEN (40+is,FILE=ptfile,access='append')
        if(n.eq.1)rewind 40+is
        i=ptsens(is,1)

```



```

        k=slsens(is,2)
        DO 120 j=jmin,jmax
            DO 110 i=imin,imax
                CALL FLDSAM(i,j,k,slsamp,field)
                WRITE (70+is,*) i,j,field
110         CONTINUE
120     CONTINUE
        ELSEIF (slsens(is,1).EQ.2) THEN
C
C         xz slice plane
C
            j=slsens(is,2)
            DO 140 i=jmin,jmax
                DO 130 k=imin,imax
                    CALL FLDSAM(i,j,k,slsamp,field)
                    WRITE (70+is,*) k,i,field
130         CONTINUE
140     CONTINUE
        ELSEIF (slsens(is,1).EQ.3) THEN
C
C         yz slice plane
C
            i=slsens(is,2)
            DO 160 k=jmin,jmax
                DO 150 j=imin,imax
                    CALL FLDSAM(i,j,k,slsamp,field)
                    WRITE (70+is,*) j,k,field
150         CONTINUE
160     CONTINUE
        ENDIF
        CLOSE (UNIT=70+is)
    ENDIF
    ENDIF
200 CONTINUE
500 RETURN
    END

```

c234567

```
C
C      SUBROUTINE READMS
C
C      INCLUDE 'main.h'
C      INTEGER iarg,line,iargc,icount
C      INTEGER i,j,k,m
C      REAL final
C      CHARACTER*30 mshfil
C      LOGICAL*1 mtype
C      REAL x,y,z,radius,rtest
C
C*****
C
C      Local variable dictionary
C
C      i=cell coordinate number in x direction
C      iarg=indicates if an argument was typed on command line
C          (i.e. if the mesh file name was supplied on the command
C          line)
C      iargc=C intrinsic function to get the number of arguments
C          typed on the command line
C      j=cell coordinate number in y direction
C      k=cell coordinate number in z direction
C      line=loop counter used for reading the mesh file (indicates
C          the current line number)
C      m=loop counter used for reading id material types
C      mshfil=mesh file name to read
C
C*****
C
C      This subroutines "builds" the object(s) of interest by
C      setting the material type id numbers and then packing them
C      in bits into an idtype array. This subroutine offers the
C      capability to read a mesh file generated by the Anastasia
C      mesh generator or to build the object by hand using hard
C      coding techniques. When reading a mesh file, the file must
C      be a binary file and must have the following form:
C
C      All lines:  i j k id1 id2 id3 id4 id5 id6
C
C      character*9 DUMC
C
C      IF (rdmesh) GO TO 100
C
C-----
C
C      This section is where you would insert your lines of code to
C      construct a particular object or objects.
C
C      radius=.19 + delx*.1
C      xc=0.5*nx1*delx
C      yc=0.5*ny1*dely
C      zc=0.5*nz1*delz
C      mtype=1
C      DO 30 k=1,nz1
C          z=(k-0.5)*delz-zc
C          DO 20 j=1,ny1
C              y=(j-0.5)*dely-yc
```

```

DO 10 i=1,nx1
  x=(i-0.5)*delx-xc
  rtest=SQRT(x*x+y*y+z*z)
  IF (rtest.LE.radius) THEN
    id1(i,j,k)=mtype
    id1(i,j,k+1)=mtype
    id1(i,j+1,k+1)=mtype
    id1(i,j+1,k)=mtype
    id2(i,j,k)=mtype
    id2(i+1,j,k)=mtype
    id2(i+1,j,k+1)=mtype
    id2(i,j,k+1)=mtype
    id3(i,j,k)=mtype
    id3(i,j+1,k)=mtype
    id3(i+1,j+1,k)=mtype
    id3(i,j+1,k)=mtype
  ENDIF
10  CONTINUE
20  CONTINUE
30  CONTINUE
  OPEN (47,FILE='sphere.cla')
  REWIND 47
  DO 35 i = 1, 29
35  READ(47,*)

    icount = 0
    DO 55 k=1,nz
      DO 55 j=1,ny
        DO 55 i=1,nx
          IF ((id1(i,j,k).GT.0).OR.(id2(i,j,k).GT.0).OR.
$          (id3(i,j,k).GT.0)) icount = icount + 1
55  CONTINUE

    WRITE(47,*)'grid_data' ,icount
    WRITE(47,*)
    DO 60 k=1,nz
      DO 50 j=1,ny
        DO 40 i=1,nx
          IF ((id1(i,j,k).GT.0).OR.(id2(i,j,k).GT.0).OR.
$          (id3(i,j,k).GT.0)) WRITE (47,777) i,j,k,id1(i,j,k),
$          id2(i,j,k),id3(i,j,k)
777  FORMAT(T2,I3,1X,I3,1X,I3,1X,I1,1X,I1,1X,I1)
40  CONTINUE
50  CONTINUE
60  CONTINUE

    WRITE(47,*)
    WRITE(47,*)'end_grid_data'
    WRITE(47,*)'end_time'
    CLOSE (UNIT=47)
    GO TO 2000

```

---

```

C 100 CONTINUE

```

```

C
C This section is used for reading mesh files.
C
C Input Geometry
C

```

```

iarg=iargc()
IF (iarg .EQ. 1) THEN
  CALL GETARG (iarg, mshfil)
ELSE
  WRITE (*,*) ' '
  WRITE (*,*) 'Enter the name of the mesh file >>'
  mshfil = 'missile9_a_cla'
  READ (5,1000) mshfil
C 1000  FORMAT (A30)
ENDIF
OPEN (53,FILE=mshfil)
REWIND 53

DO 520 line = 1,29
520  READ(53,*)
  READ(53,*)DUMC,final
  READ(53,*)

  DO 500 line=1,final
    READ (53,*,Err=510) i,j,k,(id(m),m=1,3)
    id1(i,j,k)=id(1)
    id2(i,j,k)=id(2)
    id3(i,j,k)=id(3)
    id4(i,j,k)=1
    id5(i,j,k)=1
    id6(i,j,k)=1
    if(mod(line,10000) .eq. 0)write(*,*)'line ',line,' of ',final
500  CONTINUE
    write(*,*)'finished reading mesh'
    GOTO 2000
510  write(*,*)'Error reading mesh file !!!!!!!!!!!!!!!!!!!!!!!'
    write(*,*)'Terminating code run.'

C-----
2000 RETURN
END

```

```

C      SUBROUTINE PULSE
C      INCLUDE 'main.h'

C      This subroutine sets up the source pulse function and all
C      of the necessary parameters for its specification.
C
C      First set up a Gaussian pulse
C
C      IF (gauss) THEN
C        IF (tspec) THEN
C
C          Time specifications
C
C          IF (rise) THEN
C            tau0=trise/1.1928343
C          ELSE
C            tau0=0.5*pwidth/SQRT(-LOG(ap))
C          ENDIF
C
C          Full-width half-maximum pulse width
C
C          tfwhm=2.0*tau0*SQRT(-LOG(0.5))
C          nfwhm=NINT(tfwhm/delt)
C
C          Half-power pulse width
C
C          thp=2.0*tau0*SQRT(-LOG(1.0/SQRT(2.0)))
C          nhp=NINT(thp/delt)
C
C          1/e pulse width
C
C          te=2.0*tau0*SQRT(-LOG(1.0/e))
C          nte=NINT(te/delt)
C
C          10%-90% rise time
C
C          trise=tau0*1.1928343
C          nrise=NINT(trise/delt)
C
C          Offset time delay
C
C          toff=4.0417348*tau0
C          ntoff=NINT(toff/delt)
C
C          Truncation time pulse width
C
C          ttrun=2.0*toff
C          ntrun=NINT(ttrun/delt)
C
C          Upper frequency limit
C
C          fup=c/(10.0*AMAX1(delx,dely,delz))
C          wup=twopi*fup
C
C          80 dB frequency limit
C
C          f80=(SQRT(36.841361)/tau0)/twopi

```

```

ELSEIF (fspec) THEN
C      wup=twopi*fup
      tau0=2.0/wup*SQRT(-LOG(10.0**(-adb/20.0)))
C
C      Full-width half-maximum pulse width
C
      tfwhm=2.0*tau0*SQRT(-LOG(0.5))
      nfwhm=NINT(tfwhm/delt)
C
C      Half-power pulse width
C
      thp=2.0*tau0*SQRT(-LOG(1.0/SQRT(2.0)))
      nhp=NINT(thp/delt)
C
C      1/e pulse width
C
      te=2.0*tau0*SQRT(-LOG(1.0/e))
      nte=NINT(te/delt)
C
C      10%-90% rise time
C
      trise=tau0*1.1928343
      nrise=NINT(trise/delt)
C
C      Offset time delay
C
      toff=4.0417348*tau0
      ntoff=NINT(toff/delt)
C
C      Truncation time pulse width
C
      ttrun=2.0*toff
      ntrun=NINT(ttrun/delt)
      gdb=20.0*ALOG10(EXP(-(tau0*wup/2.0)**2))
ENDIF
ELSEIF (banlim) THEN
C
C      Bandlimited pulse
C
      IF (tspec) THEN
C
C          Time specifications
C
          IF (rise) THEN
              tau0=trise/1.1928343
          ELSE
              tau0=0.5*pwidth/SQRT(-LOG(ap))
          ENDIF
C
C          Full-width half-maximum pulse width
C
          tfwhm=2.0*tau0*SQRT(-LOG(0.5))
          nfwhm=NINT(tfwhm/delt)
C
C          Half-power pulse width
C
          thp=2.0*tau0*SQRT(-LOG(1.0/SQRT(2.0)))

```

```

      nhp=NINT(thp/delt)
C
C      1/e pulse width
C
      te=2.0*tau0*SQRT(-LOG(1.0/e))
      nte=NINT(te/delt)
C
C      10%-90% rise time
C
      trise=tau0*1.1928343
      nrise=NINT(trise/delt)
C
C      Offset time delay
C
      toff=4.0417348*tau0
      ntoff=NINT(toff/delt)
C
C      Truncation time pulse width
C
      ttrun=2.0*toff
      ntrun=NINT(ttrun/delt)
C
C      Upper frequency limit
C
      fup=c/(10.0*AMAX1(delx,dely,delz))
      wup=twopi*fup
      flow=0.0
      w0=0.5*wup
C
C      80 dB frequency limit
C
      f80=(w0+SQRT(36.841361)/tau0)/twopi
      ELSEIF (fspec) THEN
C
C      Frequency specification
C
C      Center frequency
C
      f0=flow+0.5*bw
      w0=twopi*f0
C
C      Upper frequency limit
C
      fup=flow+bw
C*****
C      BOB
C      bw = bandwidth in Hz
C      f0 = center frequency in Hz
C
C      bw = 1.0e9
C      fup=bw/2.0
C      wup=twopi*fup
C      f0=.6e9
C      w0=twopi*f0
C*****
      bw=16.0e9

```

```

fup=8.1e9
wup=twopi*fup
f0=0.0
w0=twopi*f0

tau0=2.0/wup*SQRT(-4.0*LOG(10.0**(-adb/20.0)))
C
C Full-width half-maximum pulse width
C
tfwhm=2.0*tau0*SQRT(-LOG(0.5))
nfwhm=NINT(tfwhm/delt)
C
C Half-power pulse width
C
thp=2.0*tau0*SQRT(-LOG(1.0/SQRT(2.0)))
nhp=NINT(thp/delt)
C
C 1/e pulse width
C
te=2.0*tau0*SQRT(-LOG(1.0/e))
nte=NINT(te/delt)
C
C 10%-90% rise time
C
trise=tau0*1.1928343
nrise=NINT(trise/delt)
C
C Offset time delay
C
C toff=4.0417348*tau0
C*****
C toff=12.0*tau0
C*****
C ntoff=NINT(toff/delt)
C
C Truncation time pulse width
C
ttrun=2.0*toff
ntrun=NINT(ttrun/delt)
bldb=20.0*ALOG10(EXP(-(tau0*wup/2.0)**2))
ENDIF
ELSEIF (hypsec) THEN
C
C Hyperbolic secant pulse
C
C Time specifications
C
x1=1.0/ah+1.0/ah*SQRT(1.0-ah*ah)
tau0=pwidth/(2.0*LOG(x1))
C
C Full-width half-maximum pulse width
C
tfwhm=2.0*tau0*LOG(3.7320508)
nfwhm=NINT(tfwhm/delt)
C
C Half-power pulse width
C

```

```

thp=2.0*tau0*LOG(SQRT(2.0)+1.0)
nhp=NINT(thp/delt)
c
c 1/e pulse width
c
c te=2.0*tau0*LOG(5.2459401)
c nte=NINT(te/delt)
c
c 10%-90% rise time
c
c trise=2.5260775*tau0
c nrise=NINT(trise/delt)
c
c Truncation time pulse width
c
c ttrun=33.622486*tau0
c ntrun=NINT(ttrun/delt)
c
c Offset time delay
c
c toff=0.5*ttrun
c ntoff=NINT(toff/delt)
c
ELSEIF (rsine) THEN
c
c Ramped sinusoid function
c
c nper=NINT(1.0/f0/delt)
c nrise=cycles*nper
c trise=nrise*delt
c tau0=trise/1.1928343
c w0=twopi*f0
c ttrun=tsteps*delt
c ELSEIF (step) THEN
c tau0=trise/1.1928343
c
c Half-width half-maximum pulse width
c
c tfwhm=tau0*SQRT(-LOG(0.5))
c nfwhm=NINT(tfwhm/delt)
c
c Half-power pulse width
c
c thp=tau0*SQRT(-LOG(1.0/SQRT(2.0)))
c nhp=NINT(thp/delt)
c
c 1/e pulse width
c
c te=tau0*SQRT(-LOG(1.0/e))
c nte=NINT(te/delt)
c
c 10%-90% rise time
c
c nrise=NINT(trise/delt)
c
c Offset time delay
c
c toff=0.0

```

```

        ntoff=0
C      Truncation time pulse width
C
C      ttrun=(tsteps-1)*delt
        ntrun=tsteps
C
C      Upper frequency limit
C
C      fup=c/(10.0*AMAX1(delx,dely,delz))
        wup=twopi*fup
C
C      80 dB frequency limit
C
        f80=(SQRT(36.841361)/tau0)/twopi
ENDIF
tau0i=1.0/tau0
RETURN
END

```

c234567

```
c      SUBROUTINE PSRCE
c
c      INCLUDE 'main.h'
c
c      This subroutine updates the point source field component
c      The feed type is a hard source defined by the electric
c      field at a particular location being defined as function
c      of time.
c
c      tprime=time
c      INCLUDE 'source.h'
c      IF (fdtype.EQ.'x') exscat(iptsrc,jptsrc,kptsrc)=source
c      IF (fdtype.EQ.'y') eyscat(iptsrc,jptsrc,kptsrc)=source
c      IF (fdtype.EQ.'z') ezscat(iptsrc,jptsrc,kptsrc)=source
c      RETURN
c      END
```

c234567

```
      real freq(500), ampl(500)

      open(10, file='source.f')
      rewind 10
      open(20, file='source2.f')
      rewind 20

10    do 10 i=1,500
        read(10,*)freq(i), ampl(i)

20    do 20 i=500,1,-1
        write(20,*)-freq(i), ampl(i)

30    do 30 i=1,500
        write(20,*)freq(i), ampl(i)

      close(10)
      close(20)

      stop
      end
```

```
c      main.h
c234567
c      This file is the main include file for TEMAC3D.
c      It includes all of the other required header (.h) files.
c
      IMPLICIT NONE
      INCLUDE 'constants.h'
      INCLUDE 'setup.h'
      INCLUDE 'variables.h'
      INCLUDE 'sources.h'
      INCLUDE 'liao.h'
      INCLUDE 'sensors.h'
      INCLUDE 'farfld.h'
```

```

c234567
c      Include file liao.h
c
c      This file contains all of the definitions for
c      variables used in the Liao Outer Radiation
c      Boundary Condition.
c
c*****
c
c      Variable dictionary
c
c      bcx=constant multiplier for boundary condition equations
c      involving deltax
c      bcy=constant multiplier for boundary condition equations
c      involving deltay
c      bcz=constant multiplier for boundary condition equations
c      involving deltax
c      exbakx=variable for storing past time values of ex field
c      components at j=1 and j=ny boundaries
c      exbakz=variable for storing past time values of ex field
c      components at k=1 and k=nz boundaries
c      eybakx=variable for storing past time values of ey field
c      components at i=1 and i=nx boundaries
c      eybakz=variable for storing past time values of ey field
c      components at k=1 and k=nz boundaries
c      ezbakx=variable for storing past time values of ez field
c      components at i=1 and i=nx boundaries
c      ezbakz=variable for storing past time values of ez field
c      components at j=1 and j=ny boundaries
c
c*****
c
c      REAL*8 bcx,bcy,bcz,eybakx,ezbakx,
c      $exbakz,eybakz,exbakz,eybakz
c      COMMON/LIAOBC/bcx(0:2*order+1,0:order),bcy(0:2*order+1,0:order),
c      $bcz(0:2*order+1,0:order),eybakx(0:2,0:order,0:ny-1,0:nz-1),
c      $ezbakx(0:2,0:order,0:ny-1,0:nz-1),
c      $exbakz(0:2,0:order,0:nx-1,0:nz-1),
c      $eybakz(0:2,0:order,0:nx-1,0:ny-1),
c      $ezbakz(0:2,0:order,0:nx-1,0:ny-1)

```

c234567

```
C
      SUBROUTINE LIAO
C
      INCLUDE 'main.h'
C
      INTEGER i,j,k
C
C*****
C
C      Local variable dictionary
C
C      i=cell coordinate number in x direction
C      j=cell coordinate number in y direction
C      k=cell coordinate number in z direction
C      l=loop counter to sum over field components spatially
C      m=loop counter to sum over field components back in
C      time
C      m2=index into field arrays determining which time step
C      information to update
C      nb=index into field arrays determining which time step
C      information to use
C
C*****
C
C      Apply Liao ORBC to ey components at x=0 and
C      x=nx1*delx faces
C
      DO 40 k=2,nz1
        DO 30 j=1,ny1
          eybakx(1,1,j,k)=eybakx(1,1,j,k)
          eybakx(2,1,j,k)=eybakx(2,1,j,k)
          eybakx(1,2,j,k)=eybakx(1,2,j,k)+
          $bcx(1,1)*eyscat(1,j,k)+bcx(2,1)*eyscat(2,j,k)+
          $bcx(3,1)*eyscat(3,j,k)
          eybakx(2,2,j,k)=eybakx(2,2,j,k)+
          $bcx(1,1)*eyscat(nx,j,k)+bcx(2,1)*eyscat(nx-1,j,k)+
          $bcx(3,1)*eyscat(nx-2,j,k)
          eybakx(1,2,j,k)=bcx(1,2)*eyscat(1,j,k)+
          $bcx(2,2)*eyscat(2,j,k)+bcx(3,2)*eyscat(3,j,k)+
          $bcx(4,2)*eyscat(4,j,k)+bcx(5,2)*eyscat(5,j,k)
          eybakx(2,2,j,k)=bcx(1,2)*eyscat(nx,j,k)+
          $bcx(2,2)*eyscat(nx-1,j,k)+bcx(3,2)*eyscat(nx-2,j,k)+
          $bcx(4,2)*eyscat(nx-3,j,k)+bcx(5,2)*eyscat(nx-4,j,k)
        30    CONTINUE
      40    CONTINUE
C
C      Apply Liao ORBC to ez components at x=0 and
C      x=nx1*delx faces
C
      DO 90 k=1,nz1
        DO 80 j=2,ny1
          ezbakx(1,1,j,k)=ezbakx(1,1,j,k)
          ezbakx(2,1,j,k)=ezbakx(2,1,j,k)
          ezbakx(1,2,j,k)=ezbakx(1,2,j,k)+
          $bcx(1,1)*ezscat(1,j,k)+bcx(2,1)*ezscat(2,j,k)+
          $bcx(3,1)*ezscat(3,j,k)
          ezbakx(2,1,j,k)=ezbakx(2,2,j,k)+
          $bcx(1,2)*ezscat(1,j,k)+bcx(2,2)*ezscat(2,j,k)+
          $bcx(3,2)*ezscat(3,j,k)+bcx(4,2)*ezscat(4,j,k)+
          $bcx(5,2)*ezscat(5,j,k)
```

```

$bcx(1,1)*ezscat(nx,j,k)+bcx(2,1)*ezscat(nx-1,j,k)+
$bcx(3,1)*ezscat(nx-2,j,k)
    ezbakx(1,2,j,k)=bcx(1,2)*ezscat(1,j,k)+
$bcx(2,2)*ezscat(2,j,k)+bcx(3,2)*ezscat(3,j,k)+
$bcx(4,2)*ezscat(4,j,k)+bcx(5,2)*ezscat(5,j,k)
    ezbakx(2,2,j,k)=bcx(1,2)*ezscat(nx,j,k)+
$bcx(2,2)*ezscat(nx-1,j,k)+bcx(3,2)*ezscat(nx-2,j,k)+
$bcx(4,2)*ezscat(nx-3,j,k)+bcx(5,2)*ezscat(nx-4,j,k)
80    CONTINUE
90    CONTINUE

c
c    Apply Liao ORBC to ex components at y=0 and
c    y=ny1*dely faces
c
    DO 140 k=2,nz1
        DO 130 i=1,nx1
            exscat(i,1,k)=exbaky(1,1,i,k)
            exscat(i,ny,k)=exbaky(2,1,i,k)
            exbaky(1,1,i,k)=exbaky(1,2,i,k)+
$bcy(1,1)*exscat(i,1,k)+bcy(2,1)*exscat(i,2,k)+
$bcy(3,1)*exscat(i,3,k)
            exbaky(2,1,i,k)=exbaky(2,2,i,k)+
$bcy(1,1)*exscat(i,ny,k)+bcy(2,1)*exscat(i,ny-1,k)+
$bcy(3,1)*exscat(i,ny-2,k)
            exbaky(1,2,i,k)=bcy(1,2)*exscat(i,1,k)+
$bcy(2,2)*exscat(i,2,k)+bcy(3,2)*exscat(i,3,k)+
$bcy(4,2)*exscat(i,4,k)+bcy(5,2)*exscat(i,5,k)
            exbaky(2,2,i,k)=bcy(1,2)*exscat(i,ny,k)+
$bcy(2,2)*exscat(i,ny-1,k)+bcy(3,2)*exscat(i,ny-2,k)+
$bcy(4,2)*exscat(i,ny-3,k)+bcy(5,2)*exscat(i,ny-4,k)
130    CONTINUE
140    CONTINUE

c
c    Apply Liao ORBC to ez components at y=0 and
c    y=ny1*dely faces
c
    DO 190 k=1,nz1
        DO 180 i=2,nx1
            ezscat(i,1,k)=ezbaky(1,1,i,k)
            ezscat(i,ny,k)=ezbaky(2,1,i,k)
            ezbakx(1,1,i,k)=ezbaky(1,2,i,k)+
$bcy(1,1)*ezscat(i,1,k)+bcy(2,1)*ezscat(i,2,k)+
$bcy(3,1)*ezscat(i,3,k)
            ezbakx(2,1,i,k)=ezbaky(2,2,i,k)+
$bcy(1,1)*ezscat(i,ny,k)+bcy(2,1)*ezscat(i,ny-1,k)+
$bcy(3,1)*ezscat(i,ny-2,k)
            ezbakx(1,2,i,k)=bcy(1,2)*ezscat(i,1,k)+
$bcy(2,2)*ezscat(i,2,k)+bcy(3,2)*ezscat(i,3,k)+
$bcy(4,2)*ezscat(i,4,k)+bcy(5,2)*ezscat(i,5,k)
            ezbakx(2,2,i,k)=bcy(1,2)*ezscat(i,ny,k)+
$bcy(2,2)*ezscat(i,ny-1,k)+bcy(3,2)*ezscat(i,ny-2,k)+
$bcy(4,2)*ezscat(i,ny-3,k)+bcy(5,2)*ezscat(i,ny-4,k)
180    CONTINUE
190    CONTINUE

c
c    Apply Liao ORBC to ex components at z=0 and
c    z=nz1*delz faces
c

```

```

DO 240 j=2,ny1
  DO 230 i=1,nx1
    exscat(i,j,1)=exbakz(1,1,i,j)
    exscat(i,j,nz)=exbakz(2,1,i,j)
    exbakz(1,1,i,j)=exbakz(1,2,i,j)+
$bcz(1,1)*exscat(i,j,1)+bcz(2,1)*exscat(i,j,2)+
$bcz(3,1)*exscat(i,j,3)
    exbakz(2,1,i,j)=exbakz(2,2,i,j)+
$bcz(1,1)*exscat(i,j,nz)+bcz(2,1)*exscat(i,j,nz-1)+
$bcz(3,1)*exscat(i,j,nz-2)
    exbakz(1,2,i,j)=bcz(1,2)*exscat(i,j,1)+
$bcz(2,2)*exscat(i,j,2)+bcz(3,2)*exscat(i,j,3)+
$bcz(4,2)*exscat(i,j,4)+bcz(5,2)*exscat(i,j,5)
    exbakz(2,2,i,j)=bcz(1,2)*exscat(i,j,nz)+
$bcz(2,2)*exscat(i,j,nz-1)+bcz(3,2)*exscat(i,j,nz-2)+
$bcz(4,2)*exscat(i,j,nz-3)+bcz(5,2)*exscat(i,j,nz-4)
230    CONTINUE
240    CONTINUE
c
c    Apply Liao ORBC to ey components at z=0 and
c    z=nz1*delz faces
c
DO 290 j=1,ny1
  DO 280 i=2,nx1
    eyscat(i,j,1)=eybakz(1,1,i,j)
    eyscat(i,j,nz)=eybakz(2,1,i,j)
    eybakz(1,1,i,j)=eybakz(1,2,i,j)+
$bcz(1,1)*eyscat(i,j,1)+bcz(2,1)*eyscat(i,j,2)+
$bcz(3,1)*eyscat(i,j,3)
    eybakz(2,1,i,j)=eybakz(2,2,i,j)+
$bcz(1,1)*eyscat(i,j,nz)+bcz(2,1)*eyscat(i,j,nz-1)+
$bcz(3,1)*eyscat(i,j,nz-2)
    eybakz(1,2,i,j)=bcz(1,2)*eyscat(i,j,1)+
$bcz(2,2)*eyscat(i,j,2)+bcz(3,2)*eyscat(i,j,3)+
$bcz(4,2)*eyscat(i,j,4)+bcz(5,2)*eyscat(i,j,5)
    eybakz(2,2,i,j)=bcz(1,2)*eyscat(i,j,nz)+
$bcz(2,2)*eyscat(i,j,nz-1)+bcz(3,2)*eyscat(i,j,nz-2)+
$bcz(4,2)*eyscat(i,j,nz-3)+bcz(5,2)*eyscat(i,j,nz-4)
280    CONTINUE
      290    CONTINUE
    RETURN
  END

```

c234567

```
c
c      SUBROUTINE INITFF
c
c      INCLUDE 'main.h'
c      INTEGER i,j,k,iin,jin,kin
c      INTEGER mtot
c      REAL x,y,z,rfx,rfy,rfz,rhatx,rhaty,rhatz,
c      $xhat,yhat,zhat,rdrhat,xzhat,yzhat
c
c      This subroutine initializes the far-field transformation.
c
c      *****
c
c      Local variable dictionary
c
c      i=cell coordinate number in x direction
c      iin=index number into retarded time delay arrays for
c          integration surface with unit normal +/- a_x
c      j=cell coordinate number in y direction
c      jin=index number into retarded time delay arrays for
c          integration surface with unit normal +/- a_y
c      k=cell coordinate number in z direction
c      kin=index number into retarded time delay arrays for
c          integration surface with unit normal +/- a_z
c      mtot=the total number of time bins required for the far field
c          vector potentials
c      rdrhat=r*rhat (the dot product of the position vector for the
c          cell being integrated over with the unit vector in the
c          direction of the far field observation point
c      rf=maximum distance of rfx, rfy, rfz
c      rfx=maximum distance from center of problem space to a cell
c          on the integration surface with unit normal a_x.
c      rfy=maximum distance from center of problem space to a cell
c          on the integration surface with unit normal a_y.
c      rfz=maximum distance from center of problem space to a cell
c          on the integration surface with unit normal a_z.
c      rhatx=x component of unit vector in direction of far field
c          observation point
c      rhaty=y component of unit vector in direction of far field
c          observation point
c      rhatz=z component of unit vector in direction of far field
c          observation point
c      x=x coordinate of a cell on the integration surface
c      xhat=x*rhatx
c      xzhat=x*rhatx+z*rhatz
c      y=y coordinate of a cell on the integration surface
c      yhat=y*rhaty
c      yzhat=y*rhaty+z*rhatz
c      z=z coordinate of a cell on the integration surface
c      zhat=z*rhatz
c
c      *****
c
c      Compute upper and lower indices of far field integration
c      surface
c
c      iup=nx-5
```

```

      jup=ny-5
      kup=nz-5
      ilow=5
      jlow=5
      klow=5
C
C      Compute distance terms to determine maximum distance from
C      center of space to a cell on the integration surface
C
      rfx=SQRT(((iup-1)*delx-xc)**2+((jup-0.5)*dely-yc)**2+
      $((kup-0.5)*delz-zc)**2)
      rfy=SQRT(((iup-0.5)*delx-xc)**2+((jup-1)*dely-yc)**2+
      $((kup-0.5)*delz-zc)**2)
      rfz=SQRT(((iup-0.5)*delx-xc)**2+((jup-0.5)*dely-yc)**2+
      $((kup-1)*delz-zc)**2)
C
C      Take the maximum of the three distance terms
C
      rf=AMAX1(rfx,rfy,rfz)
C
C      Check the number of time bins needed for the transformation
C      against the maximum number of time bins allotted.
C
      mtot=tsteps+INT(rf/(c*delt))+10
      IF (mtot.GT.nmax) THEN
        WRITE (15,*) ' '
        WRITE (15,*) 'Error! The number of time bins for the'
        WRITE (15,*) 'far field vector potential arrays is too'
        WRITE (15,*) 'small. Please set parameter MMAX in file'
        WRITE (15,*) 'setup.h to be >= ',mtot
        WRITE (15,*) 'Execution halted.'
        errflg=true
      ENDIF
C
C      Initialize the far field vector potential pointer arrays
C
      inuwx(1)=5
      inuwx(2)=6
      inuwx(3)=2
      inuwx(4)=3
      inuwy(1)=4
      inuwy(2)=6
      inuwy(3)=1
      inuwy(4)=3
      inuwz(1)=4
      inuwz(2)=5
      inuwz(3)=1
      inuwz(4)=2
      fpcdt=1.0/(4.0*pi*c*delt)
C
C      Compute the unit vector in the direction of the far-field
C      observation point.
C
      rhata=sin(theta*degrad)*cos(phi*degrad)
      rhaty=sin(theta*degrad)*sin(phi*degrad)
      rhatz=cos(theta*degrad)
C
C      Now fill the time delay arrays

```

```

c
c      Start with faces with unit normal +/- a_x.
c
      x=(ilow-1)*delx-xc
      DO 30 i=1,2
            xhat=x*rhatx
            DO 20 k=klow,kup
                  kin=k-klow+1
                  z=(k-0.5)*delz-zc
                  zhat=z*rhatsz
                  xzhat=xhat+zhat
                  DO 10 j=jlow,jup
                        jin=j-jlow+1
                        y=(j-0.5)*dely-yc
                        rdrhat=xzhat+y*rhaty
                        tretx(jin,kin,i)=(rf-rdrhat)*cinv
10          CONTINUE
20          CONTINUE
      x=(iup-1)*delx-xc
30          CONTINUE
c
c      Next compute delays for faces with unit normal +/- a_y.
c
      y=(jlow-1)*dely-yc
      DO 60 j=1,2
            yhat=y*rhaty
            DO 50 k=klow,kup
                  kin=k-klow+1
                  z=(k-0.5)*delz-zc
                  zhat=z*rhatsz
                  yzhat=yhat+zhat
                  DO 40 i=ilow,iup
                        iin=i-ilow+1
                        x=(i-0.5)*delx-xc
                        rdrhat=yzhat+x*rhatx
                        tetry(iin,kin,j)=(rf-rdrhat)*cinv
40          CONTINUE
50          CONTINUE
      y=(jup-1)*dely-yc
60          CONTINUE
c
c      Finally compute delays for faces with unit normal +/- a_z.
c
      z=(klow-1)*delz-zc
      DO 90 k=1,2
            zhat=z*rhatsz
            DO 80 j=jlow,jup
                  jin=j-jlow+1
                  y=(j-0.5)*dely-yc
                  yhat=y*rhaty
                  yzhat=yhat+zhat
                  DO 70 i=ilow,iup
                        iin=i-ilow+1
                        x=(i-0.5)*delx-xc
                        rdrhat=yzhat+x*rhatx
                        tretz(iin,jin,k)=(rf-rdrhat)*cinv
70          CONTINUE
80          CONTINUE

```

```
      z=(kup-1)*delz-zc
90    CONTINUE
      RETURN
      END
```

```

C
C      SUBROUTINE HOLL3D
C
C      THIS SUBROUTINE HOLLOWS OUT THREE DIMENSIONAL FDTD OBJECTS
C
C      AUTHOR:  JOHN H. BEGGS
C      DATE:    7/16/92
C
C      SUBROUTINE HOLL3D
C      INCLUDE 'main.h'
C      LOGICAL*1 iprod, jprod, kprod
C      INTEGER i, j, k
C
C      DO 9 k=2,nz1
C          DO 8 j=2,ny1
C              DO 7 i=2,nx1
C                  iprod=id1(i,j+1,k)*id1(i,j-1,k)*id1(i,j,k-1)*
C                  $id1(i,j,k+1)*id2(i,j,k)*id2(i+1,j,k)*id2(i,j-1,k)*
C                  $id2(i+1,j-1,k)*id3(i,j,k)*id3(i+1,j,k)*
C                  $id3(i,j,k-1)*id3(i+1,j,k-1)
C                  iprod=iprod*id2(i,j,k+1)*id2(i,j-1,k+1)*id2(i,j,k-1)*
C                  $id2(i,j-1,k-1)*id2(i+1,j,k+1)*id2(i+1,j-1,k+1)*
C                  $id2(i+1,j,k-1)*id2(i+1,j-1,k-1)
C                  iprod=iprod*id3(i,j+1,k)*id3(i,j-1,k)*id3(i,j-1,k-1)*
C                  $id3(i,j+1,k-1)*id3(i+1,j+1,k)*id3(i+1,j-1,k)*
C                  $id3(i+1,j-1,k-1)*id3(i+1,j+1,k-1)
C                  jprod=id2(i+1,j,k)*id2(i-1,j,k)*id2(i,j,k-1)*
C                  $id2(i,j,k+1)*id1(i,j,k)*id1(i,j+1,k)*id1(i-1,j,k)*
C                  $id1(i-1,j+1,k)*id3(i,j,k)*id3(i,j+1,k)*id3(i,j,k-1)*
C                  $id3(i,j+1,k-1)
C                  jprod=jprod*id1(i,j,k+1)*id1(i-1,j,k+1)*id1(i,j,k-1)*
C                  $id1(i-1,j,k-1)*id1(i,j+1,k+1)*id1(i-1,j+1,k+1)*
C                  $id1(i,j+1,k-1)*id1(i-1,j+1,k-1)
C                  jprod=jprod*id3(i+1,j,k)*id3(i-1,j,k)*id3(i+1,j,k-1)*
C                  $id3(i-1,j,k-1)*id3(i+1,j+1,k)*id3(i-1,j+1,k)*
C                  $id3(i+1,j+1,k-1)*id3(i-1,j+1,k-1)
C                  kprod=id3(i-1,j,k)*id3(i+1,j,k)*id3(i,j-1,k)*
C                  $id3(i,j+1,k)*id1(i,j,k)*id1(i,j,k+1)*id1(i-1,j,k)*
C                  $id1(i-1,j,k+1)*id2(i,j,k)*id2(i,j,k+1)*id2(i,j-1,k)*
C                  $id2(i,j-1,k+1)
C                  kprod=kprod*id1(i,j+1,k)*id1(i-1,j+1,k)*id1(i-1,j-1,k)*
C                  $id1(i,j-1,k)*id1(i,j+1,k+1)*id1(i-1,j+1,k+1)*
C                  $id1(i-1,j-1,k+1)*id1(i,j-1,k+1)
C                  kprod=kprod*id2(i+1,j,k)*id2(i-1,j,k)*id2(i-1,j-1,k)*
C                  $id2(i+1,j-1,k)*id2(i+1,j,k+1)*id2(i-1,j,k+1)*
C                  $id2(i-1,j-1,k+1)*id2(i+1,j-1,k+1)
C                  IF (iprod.NE.0) id1(i,j,k)=-1
C                  IF (jprod.NE.0) id2(i,j,k)=-1
C                  IF (kprod.NE.0) id3(i,j,k)=-1
C              7      CONTINUE
C          8      CONTINUE
C      9      CONTINUE
C      DO 12 k=1,nz
C          DO 11 j=1,ny
C              DO 10 i=1,nx
C                  IF (id1(i,j,k).EQ.-1) id1(i,j,k)=0
C                      IF (id2(i,j,k).EQ.-1) id2(i,j,k)=0
C                          IF (id3(i,j,k).EQ.-1) id3(i,j,k)=0

```

```
10      CONTINUE
11      CONTINUE
12      CONTINUE
      RETURN
      END
```

c234567

```
c
c      SUBROUTINE UPDHXS
c
c      This subroutine updates the x component of scattered
c      magnetic field.
c
c      INCLUDE 'main.h'
c      INTEGER i,j,k
c
c*****
c
c      Local variable dictionary
c
c      i=cell coordinate number in x direction
c      j=cell coordinate number in y direction
c      k=cell coordinate number in z direction
c*****
c
c      DO 30 k=1,nz1
c        DO 20 j=1,ny1
c          DO 10 i=2,nx1
c
c            Get the material type
c
c            IF (id4(i,j,k).LT.2) THEN
c
c              Non-magnetic material update equation
c
c              hxscat(i,j,k)=hxscat(i,j,k)-(ezscat(i,j+1,k)-
c                $ezscat(i,j,k))*dtomdy+
c                $(eyscat(i,j,k+1)-eyscat(i,j,k))*dtomdz
c
c            ELSE
c
c              Lossy magnetic materials
c
c              hxscat(i,j,k)=hxscat(i,j,k)*hold(id4(i,j,k))-
c                $(ezscat(i,j+1,k)-ezscat(i,j,k))*dedy(id4(i,j,k))+
c                $(eyscat(i,j,k+1)-eyscat(i,j,k))*dedz(id4(i,j,k))
c              IF (.NOT.plwave) GO TO 10
c              rdrhat=(i-1)*dxcosa+(j-0.5)*dycosb+(k-0.5)*dzcosg
c              tprime=time-rdrhat*cinv-tdelay
c              IF ((tprime.LT.ttrun).AND.(tprime.gt.0.)) THEN
c                INCLUDE 'source.h'
c                INCLUDE 'ddtsrce.h'
c                hxscat(i,j,k)=hxscat(i,j,k)-hamplx*
c                $(hinc(id4(i,j,k))*source+ddthin(id4(i,j,k))*ddtsrc)
c              ENDIF
c
c            ENDIF
c
c          CONTINUE
c        CONTINUE
c      CONTINUE
c      RETURN
c      END
c
```

```

SUBROUTINE UPDHYS
C
C   This subroutine updates the y component of scattered
C   magnetic field.
C
C   INCLUDE 'main.h'
C   INTEGER i,j,k
C
C*****
C
C   Local variable dictionary
C
C   i=cell coordinate number in x direction
C   j=cell coordinate number in y direction
C   k=cell coordinate number in z direction
C*****
C
C   DO 30 k=1,nz1
C     DO 20 j=2,ny1
C       DO 10 i=1,nx1
C
C         Get the material type
C
C         IF (id5(i,j,k).LT.2) THEN
C
C           Non-magnetic material update equation
C
C           hyscat(i,j,k)=hyscat(i,j,k)+(ezscat(i+1,j,k)-
C             $ezscat(i,j,k))*dtomdx-
C             $(exscat(i,j,k+1)-exscat(i,j,k))*dtomdz
C
C           ELSE
C
C             Lossy magnetic materials
C
C             hyscat(i,j,k)=hyscat(i,j,k)*hold(id5(i,j,k))+
C             $(ezscat(i+1,j,k)-ezscat(i,j,k))*dedx(id5(i,j,k))-
C             $(exscat(i,j,k+1)-exscat(i,j,k))*dedz(id5(i,j,k))
C             IF (.NOT.plwave) GO TO 10
C             rdrhat=(i-0.5)*dxcosa+(j-1)*dycosb+(k-0.5)*dzcosg
C             tprime=time-rdrhat*cinv-tdelay
C             IF ((tprime.LT.ttrun).AND.(tprime.gt.0.)) THEN
C               INCLUDE 'source.h'
C               INCLUDE 'ddtsrce.h'
C               hyscat(i,j,k)=hyscat(i,j,k)-hamply*
C               $(hinc(id5(i,j,k))*source+ddthin(id5(i,j,k))*ddtsrc)
C             ENDIF
C
C           ENDIF
C
C         ENDIF
C
C       10      CONTINUE
C     20      CONTINUE
C   30      CONTINUE
C   RETURN
C   END
C
SUBROUTINE UPDHZS

```

```

C      This subroutine updates the z component of scattered
C      magnetic field.
C
C      INCLUDE 'main.h'
C      INTEGER i,j,k
C
C*****
C      Local variable dictionary
C
C      i=cell coordinate number in x direction
C      j=cell coordinate number in y direction
C      k=cell coordinate number in z direction
C*****
C
C      DO 30 k=2,nz1
C        DO 20 j=1,nyl
C          DO 10 i=1,nx1
C
C            Get the material type
C
C            IF (id6(i,j,k).LT.2) THEN
C
C              Non-magnetic material update equation
C
C              hzscat(i,j,k)=hzscat(i,j,k)+(exscat(i,j+1,k)-
C                $exscat(i,j,k))*dtomdy-
C                $(eyscat(i+1,j,k)-eyscat(i,j,k))*dtomdx
C
C            ELSE
C
C              Lossy magnetic materials
C
C              hzscat(i,j,k)=hzscat(i,j,k)*hold(id6(i,j,k))+
C                $(exscat(i,j+1,k)-exscat(i,j,k))*dedy(id6(i,j,k))-
C                $(eyscat(i+1,j,k)-eyscat(i,j,k))*dedx(id6(i,j,k))
C              IF (.NOT.plwave) GO TO 10
C              rdrhat=(i-0.5)*dxcosa+(j-0.5)*dycosb+(k-1)*dzcosg
C              tprime=time-rdrhat*cinv-tdelay
C              IF ((tprime.LT.ttrun).AND.(tprime.gt.0.)) THEN
C                INCLUDE 'source.h'
C                INCLUDE 'ddtsrce.h'
C                hzscat(i,j,k)=hzscat(i,j,k)-hamplz*
C                $(hinc(id6(i,j,k))*source+ddthin(id6(i,j,k))*ddtsrc)
C              ENDIF
C
C            ENDIF
C
C          ENDIF
C
C        CONTINUE
C      CONTINUE
C    CONTINUE
C  RETURN
C  END

```

```

c234567
      SUBROUTINE FLDSAM(i,j,k,type,field)
C
      INCLUDE 'main.h'
      INTEGER i,j,k,type
      REAL field,hyincijk1,hyincijk,hzincijk,hzincijk1,exincijk
      REAL hxincijk,hxincijk1,hzincijk1,eyincijk,hyincijk1,hxincijk1
      REAL ezincijk
C
C      This subroutine samples a particular field quantity for
C      saving to point or slice sensor data file.
C
C*****
C      Local variable dictionary
C
C      field=sampled field quantity
C      i=cell coordinate number in x direction
C      j=cell coordinate number in y direction
C      k=cell coordinate number in z direction
C      type=number of field quantity to be sampled
C*****
C
C      GO TO (11,12,13,14,15,16,17,18,19,20,21,22,23,
$24,25,26,27,28,29,30,31,32,33,34,35,36,37,38,39,41,42,43,44) type
C
C      Ex scattered field
C
C      11  field=exscat(i,j,k)
C          GO TO 40
C
C      Ey scattered field
C
C      12  field=eyscat(i,j,k)
C          GO TO 40
C
C      Ez scattered field
C
C      13  field=ezscat(i,j,k)
C          GO TO 40
C
C      Hx scattered field
C
C      14  field=hxscat(i,j,k)
C          GO TO 40
C
C      Hy scattered field
C
C      15  field=hyscat(i,j,k)
C          GO TO 40
C
C      Hz scattered field
C
C      16  field=hzscat(i,j,k)
C          GO TO 40
C
C      x-directed conduction current

```

```

C
17  rdrhat=(i-0.5)*dxcosa+(j-1)*dycosb+(k-1)*dzcosg
    tprime=time-rdrhat*cinv-tdelay
    INCLUDE 'source.h'
    field=sigma(id1(i,j,k))*(exscat(i,j,k) + eamplx*source)
    GO TO 40

C
C    y-directed conduction current
C
18  rdrhat=(i-1)*dxcosa+(j-0.5)*dycosb+(k-1)*dzcosg
    tprime=time-rdrhat*cinv-tdelay
    INCLUDE 'source.h'
    field=sigma(id2(i,j,k))*(eyscat(i,j,k) + eamply*source)
    GO TO 40

C
C    z-directed conduction current
C
19  rdrhat=(i-1)*dxcosa+(j-1)*dycosb+(k-0.5)*dzcosg
    tprime=time-rdrhat*cinv-tdelay
    INCLUDE 'source.h'
    field=sigma(id3(i,j,k))*(ezscat(i,j,k) + eamplz*source)
    GO TO 40

C
C    x-directed displacement current
C
20  rdrhat=(i-.5)*dxcosa+(j-1)*dycosb+(k-1.5)*dzcosg
    tprime=time-rdrhat*cinv-tdelay
    INCLUDE 'source.h'
    hyincijk1=-hamply*source

    rdrhat=(i-.5)*dxcosa+(j-1)*dycosb+(k-.5)*dzcosg
    tprime=time-rdrhat*cinv-tdelay
    hyincijk=-hamply*source

    rdrhat=(i-.5)*dxcosa+(j-.5)*dycosb+(k-1)*dzcosg
    tprime=time-rdrhat*cinv-tdelay
    hzincijk=-hamplz*source

    rdrhat=(i-.5)*dxcosa+(j-1.5)*dycosb+(k-1)*dzcosg
    tprime=time-rdrhat*cinv-tdelay
    hzincijk=-hamplz*source

    rdrhat=(i-0.5)*dxcosa+(j-1)*dycosb+(k-1)*dzcosg
    tprime=time-rdrhat*cinv-tdelay
    exincijk=-eamplx*source

    field=(hyscat(i,j,k-1)+hyincijk1-hyscat(i,j,k)-hyincijk)*dely+
    $(hzscat(i,j,k)+hzincijk-hzscat(i,j-1,k)-hzincijk)*delz-
    $sigma(id1(i,j,k))*(exscat(i,j,k)+exincijk)
    GO TO 40

C
C    y-directed displacement current
C
21  rdrhat=(i-1)*dxcosa+(j-0.5)*dycosb+(k-0.5)*dzcosg
    tprime=time-rdrhat*cinv-tdelay
    INCLUDE 'source.h'
    hxincijk=-hamplx*source

```

```

rdrhat=(i-1)*dxcosa+(j-0.5)*dycosb+(k-1.5)*dzcosg
tprime=time-rdrhat*cinv-tdelay
hxincijkl=-hamplx*source

rdrhat=(i-1.5)*dxcosa+(j-.5)*dycosb+(k-1)*dzcosg
tprime=time-rdrhat*cinv-tdelay
hzinciljk=-hamplz*source

rdrhat=(i-.5)*dxcosa+(j-.5)*dycosb+(k-1)*dzcosg
tprime=time-rdrhat*cinv-tdelay
hzincijk=-hamplz*source

rdrhat=(i-1)*dxcosa+(j-0.5)*dycosb+(k-1)*dzcosg
tprime=time-rdrhat*cinv-tdelay
eyincijk=-eamply*source

field=(hxscat(i,j,k)+hxincijk-hxscat(i,j,k-1)-hxincijkl)*delx+
$(hzscat(i-1,j,k)+hzinciljk-hzscat(i,j,k)-hzincijk)*delz-
$sigma(id2(i,j,k))*(eyscat(i,j,k)+eyincijk)
GO TO 40

C
C      z-directed displacement current
C
22  rdrhat=(i-.5)*dxcosa+(j-1)*dycosb+(k-0.5)*dzcosg
    tprime=time-rdrhat*cinv-tdelay
    INCLUDE 'source.h'
    hyincijk=-hamply*source

    rdrhat=(i-1.5)*dxcosa+(j-1)*dycosb+(k-0.5)*dzcosg
    tprime=time-rdrhat*cinv-tdelay
    hyinciljk=-hamply*source

    rdrhat=(i-1)*dxcosa+(j-1.5)*dycosb+(k-0.5)*dzcosg
    tprime=time-rdrhat*cinv-tdelay
    hxincijlk=-hamplx*source

    rdrhat=(i-1)*dxcosa+(j-.5)*dycosb+(k-0.5)*dzcosg
    tprime=time-rdrhat*cinv-tdelay
    hxincijk=-hamplx*source

    rdrhat=(i-1)*dxcosa+(j-1)*dycosb+(k-.5)*dzcosg
    tprime=time-rdrhat*cinv-tdelay
    ezincijk=-eamplz*source

    field=(hyscat(i,j,k)+hyincijk-hyscat(i-1,j,k)-hyinciljk)*dely+
$(hxscat(i,j-1,k)+hxincijlk-hxscat(i,j,k)-hxincijk)*delx-
$sigma(id3(i,j,k))*(ezscat(i,j,k)+ezincijk)
GO TO 40

C
C      x-directed total current
C
23  rdrhat=(i-0.5)*dxcosa+(j-1)*dycosb+(k-1.5)*dzcosg
    tprime=time-rdrhat*cinv-tdelay
    hyincijkl=0.0
    IF ((tprime.LT.ttrun).AND.(tprime.GT.0.)) THEN
        INCLUDE 'source.h'
        hyincijkl=hamply*source
    ENDIF

```

```

rdrhat=(i-0.5)*dxcosa+(j-1)*dycosb+(k-.5)*dzcosg
tprime=time-rdrhat*cinv-tdelay
hyincijk=0.0
IF ((tprime.LT.ttrun).AND.(tprime.GT.0.)) THEN
    INCLUDE 'source.h'
    hyincijk=hamply*source
ENDIF

rdrhat=(i-.5)*dxcosa+(j-.5)*dycosb+(k-1)*dzcosg
tprime=time-rdrhat*cinv-tdelay
hzincijk=0.0
IF ((tprime.LT.ttrun).AND.(tprime.GT.0.)) THEN
    INCLUDE 'source.h'
    hzincijk=hamply*source
ENDIF

rdrhat=(i-.5)*dxcosa+(j-1.5)*dycosb+(k-1)*dzcosg
tprime=time-rdrhat*cinv-tdelay
hzincijlk=0.0
IF ((tprime.LT.ttrun).AND.(tprime.GT.0.)) THEN
    INCLUDE 'source.h'
    hzincijlk=hamply*source
ENDIF

field=(hyscat(i,j,k-1)+hyincijk1-hyscat(i,j,k)-hyincijk)*dely+
$(hzscat(i,j,k)+hzincijk-hzscat(i,j-1,k)-hzincijlk)*delz

```

GO TO 40

c  
c  
c

y-directed total current

24

```

rdrhat=(i-1)*dxcosa+(j-0.5)*dycosb+(k-0.5)*dzcosg
tprime=time-rdrhat*cinv-tdelay
hxincijk=0.0
IF ((tprime.LT.ttrun).AND.(tprime.GT.0.)) THEN
    INCLUDE 'source.h'
    hxincijk=hamply*source
ENDIF

```

```

rdrhat=(i-1)*dxcosa+(j-0.5)*dycosb+(k-1.5)*dzcosg
tprime=time-rdrhat*cinv-tdelay
hxincijk1=0.0
IF ((tprime.LT.ttrun).AND.(tprime.GT.0.)) THEN
    INCLUDE 'source.h'
    hxincijk1=hamply*source
ENDIF

```

```

rdrhat=(i-1.5)*dxcosa+(j-.5)*dycosb+(k-1)*dzcosg
tprime=time-rdrhat*cinv-tdelay
hzinciljk=0.0
IF ((tprime.LT.ttrun).AND.(tprime.GT.0.)) THEN
    INCLUDE 'source.h'
    hzinciljk=hamply*source
ENDIF

```

```

rdrhat=(i-.5)*dxcosa+(j-.5)*dycosb+(k-1)*dzcosg
tprime=time-rdrhat*cinv-tdelay
hzincijk=0.0

```

```

IF ((tprime.LT.ttrun).AND.(tprime.GT.0.)) THEN
  INCLUDE 'source.h'
  hzincijk=hamplz*source
ENDIF

field=(hxscat(i,j,k)+hxincijk-hxscat(i,j,k-1)-hxincijk1)*delx+
$(hzscat(i-1,j,k)+hzinciljk-hzscat(i,j,k)-hzincijk)*delz
GO TO 40

C
C   z-directed total current
C
25  rdrhat=(i-.5)*dxcosa+(j-1)*dycosb+(k-0.5)*dzcosg
    tprime=time-rdrhat*cinv-tdelay
    hyincijk=0.0
    IF ((tprime.LT.ttrun).AND.(tprime.GT.0.)) THEN
      INCLUDE 'source.h'
      hyincijk=hamply*source
    ENDIF

    rdrhat=(i-1.5)*dxcosa+(j-1)*dycosb+(k-0.5)*dzcosg
    tprime=time-rdrhat*cinv-tdelay
    hyinciljk=0.0
    IF ((tprime.LT.ttrun).AND.(tprime.GT.0.)) THEN
      INCLUDE 'source.h'
      hyinciljk=hamply*source
    ENDIF

    rdrhat=(i-1)*dxcosa+(j-1.5)*dycosb+(k-0.5)*dzcosg
    tprime=time-rdrhat*cinv-tdelay
    hxincijlk=0.0
    IF ((tprime.LT.ttrun).AND.(tprime.GT.0.)) THEN
      INCLUDE 'source.h'
      hxincijlk=hamplx*source
    ENDIF

    rdrhat=(i-1)*dxcosa+(j-.5)*dycosb+(k-0.5)*dzcosg
    tprime=time-rdrhat*cinv-tdelay
    hxincijk=0.0
    IF ((tprime.LT.ttrun).AND.(tprime.GT.0.)) THEN
      INCLUDE 'source.h'
      hxincijk=hamplx*source
    ENDIF

    field=(hyscat(i,j,k)+hyincijk-hyscat(i-1,j,k)-hyinciljk)*dely+
    $(hxscat(i,j-1,k)+hxincijlk-hxscat(i,j,k)-hxincijk)*delx

    GO TO 40

C
C   Ex total field
C
26  field=exscat(i,j,k)
    rdrhat=(i-0.5)*dxcosa+(j-1)*dycosb+(k-1)*dzcosg
    tprime=time-rdrhat*cinv-tdelay
    IF ((tprime.LT.ttrun).AND.(tprime.GT.0.)) THEN
      INCLUDE 'source.h'
      field=field + eamplx*source
    ENDIF
    GO TO 40

```

```

C
C      Ey total field
C
27      field=eyscat(i,j,k)
         rdrhat=(i-1)*dxcosa+(j-0.5)*dycosb+(k-1)*dzcosg
         tprime=time-rdrhat*cinv-tdelay
         IF ((tprime.LT.ttrun).AND.(tprime.GT.0.)) THEN
             INCLUDE 'source.h'
             field=field + eamply*source
         ENDIF
         GO TO 40
C
C      Ez total field
C
28      field=ezscat(i,j,k)
         rdrhat=(i-1)*dxcosa+(j-1)*dycosb+(k-0.5)*dzcosg
         tprime=time-rdrhat*cinv-tdelay
         IF ((tprime.LT.ttrun).AND.(tprime.GT.0.)) THEN
             INCLUDE 'source.h'
             field=field + eamplz*source
         ENDIF
         GO TO 40
C
C      Hx total field
C
29      field=hxscat(i,j,k)
         rdrhat=(i-1)*dxcosa+(j-0.5)*dycosb+(k-0.5)*dzcosg
         tprime=time-rdrhat*cinv-tdelay
         IF ((tprime.LT.ttrun).AND.(tprime.GT.0.)) THEN
             INCLUDE 'source.h'
             field=field + hamplx*source
         ENDIF
         GO TO 40
C
C      Hy total field
C
30      field=hyscat(i,j,k)
         rdrhat=(i-0.5)*dxcosa+(j-1)*dycosb+(k-0.5)*dzcosg
         tprime=time-rdrhat*cinv-tdelay
         IF ((tprime.LT.ttrun).AND.(tprime.GT.0.)) THEN
             INCLUDE 'source.h'
             field=hyscat(i,j,k) + hamply*source
         ENDIF
         GO TO 40
C
C      Hz total field
C
31      field=hzscat(i,j,k)
         rdrhat=(i-0.5)*dxcosa+(j-0.5)*dycosb+(k-1)*dzcosg
         tprime=time-rdrhat*cinv-tdelay
         IF ((tprime.LT.ttrun).AND.(tprime.GT.0.)) THEN
             INCLUDE 'source.h'
             field=hzscat(i,j,k) + hamplz*source
         ENDIF
         GO TO 40
C
C      Ex incident field

```

```

C
32  rdrhat=(i-0.5)*dxcosa+(j-1)*dycosb+(k-1)*dzcosg
    tprime=time-rdrhat*cinv-tdelay
    IF ((tprime.LT.ttrun).AND.(tprime.GT.0.)) THEN
        INCLUDE 'source.h'
        field=eamplx*source
    ENDIF
    GO TO 40

C
C  Ey incident field
C
33  rdrhat=(i-1)*dxcosa+(j-0.5)*dycosb+(k-1)*dzcosg
    tprime=time-rdrhat*cinv-tdelay
    IF ((tprime.LT.ttrun).AND.(tprime.GT.0.)) THEN
        INCLUDE 'source.h'
        field=eamply*source
    ENDIF
    GO TO 40

C
C  Ez incident field
C
34  rdrhat=(i-1)*dxcosa+(j-1)*dycosb+(k-0.5)*dzcosg
    tprime=time-rdrhat*cinv-tdelay
    IF ((tprime.LT.ttrun).AND.(tprime.GT.0.)) THEN
        INCLUDE 'source.h'
        field= eamplz*source
    ENDIF
    GO TO 40

C
C  Hx incident field
C
35  rdrhat=(i-1)*dxcosa+(j-0.5)*dycosb+(k-0.5)*dzcosg
    tprime=time-rdrhat*cinv-tdelay
    IF ((tprime.LT.ttrun).AND.(tprime.GT.0.)) THEN
        INCLUDE 'source.h'
        field=hamplx*source
    ENDIF
    GO TO 40

C
C  Hy incident field
C
36  rdrhat=(i-0.5)*dxcosa+(j-1)*dycosb+(k-0.5)*dzcosg
    tprime=time-rdrhat*cinv-tdelay
    IF ((tprime.LT.ttrun).AND.(tprime.GT.0.)) THEN
        INCLUDE 'source.h'
        field=hamply*source
    ENDIF
    GO TO 40

C
C  Hz incident field
C
37  rdrhat=(i-0.5)*dxcosa+(j-0.5)*dycosb+(k-1)*dzcosg
    tprime=time-rdrhat*cinv-tdelay
    IF ((tprime.LT.ttrun).AND.(tprime.GT.0.)) THEN
        INCLUDE 'source.h'
        field= hamplz*source
    ENDIF
    GO TO 40

```

```

C      x-directed total scattered current
C
C      38  field=(hyscat(i,j,k-1)-hyscat(i,j,k))*delz+
          $(hzscat(i,j,k)-hzscat(i,j-1,k))*dely
          GO TO 40
C
C      y-directed total scattered current
C
C      39  field=(hxscat(i,j,k)-hxscat(i,j,k-1))*delz+
          $(hzscat(i-1,j,k)-hzscat(i,j,k))*delx
          GO TO 40
C
C      z-directed total scattered current
C
C      41  field=(hyscat(i,j,k)-hyscat(i-1,j,k))*delx+
          $(hxscat(i,j-1,k)-hxscat(i,j,k))*dely
          GO TO 40
C
C      x-directed total incident current
C
C      42  rdrhat=(i-0.5)*dxcosa+(j-1)*dycosb+(k-1.5)*dzcosg
          tprime=time-rdrhat*cinv-tdelay
          hyincijkl=0.0
          IF ((tprime.LT.ttrun).AND.(tprime.GT.0.)) THEN
              INCLUDE 'source.h'
              hyincijkl=hamply*source
          ENDIF

          rdrhat=(i-0.5)*dxcosa+(j-1)*dycosb+(k-.5)*dzcosg
          tprime=time-rdrhat*cinv-tdelay
          hyincijk=0.0
          IF ((tprime.LT.ttrun).AND.(tprime.GT.0.)) THEN
              INCLUDE 'source.h'
              hyincijk=hamply*source
          ENDIF

          rdrhat=(i-.5)*dxcosa+(j-.5)*dycosb+(k-1)*dzcosg
          tprime=time-rdrhat*cinv-tdelay
          hzincijk=0.0
          IF ((tprime.LT.ttrun).AND.(tprime.GT.0.)) THEN
              INCLUDE 'source.h'
              hzincijk=hamplz*source
          ENDIF

          rdrhat=(i-.5)*dxcosa+(j-1.5)*dycosb+(k-1)*dzcosg
          tprime=time-rdrhat*cinv-tdelay
          hzincijlk=0.0
          IF ((tprime.LT.ttrun).AND.(tprime.GT.0.)) THEN
              INCLUDE 'source.h'
              hzincijlk=hamplz*source
          ENDIF

          field=(hyincijkl-hyincijk)*dely+
          $(hzincijk-hzincijlk)*delz
          GO TO 40
C
C      y-directed total incident current

```

```

c
43  rdrhat=(i-1)*dxcosa+(j-0.5)*dycosb+(k-0.5)*dzcosg
    tprime=time-rdrhat*cinv-tdelay
    hxincijk=0.0
    IF ((tprime.LT.ttrun).AND.(tprime.GT.0.)) THEN
        INCLUDE 'source.h'
        hxincijk=hamp1x*source
    ENDIF

    rdrhat=(i-1)*dxcosa+(j-0.5)*dycosb+(k-1.5)*dzcosg
    tprime=time-rdrhat*cinv-tdelay
    hxincijk1=0.0
    IF ((tprime.LT.ttrun).AND.(tprime.GT.0.)) THEN
        INCLUDE 'source.h'
        hxincijk1=hamp1x*source
    ENDIF

    rdrhat=(i-1.5)*dxcosa+(j-.5)*dycosb+(k-1)*dzcosg
    tprime=time-rdrhat*cinv-tdelay
    hzinciljk=0.0
    IF ((tprime.LT.ttrun).AND.(tprime.GT.0.)) THEN
        INCLUDE 'source.h'
        hzinciljk=hamp1z*source
    ENDIF

    rdrhat=(i-.5)*dxcosa+(j-.5)*dycosb+(k-1)*dzcosg
    tprime=time-rdrhat*cinv-tdelay
    hzincijk=0.0
    IF ((tprime.LT.ttrun).AND.(tprime.GT.0.)) THEN
        INCLUDE 'source.h'
        hzincijk=hamp1z*source
    ENDIF

    field=(hxincijk-hxincijk1)*delx+
    $(hzinciljk-hzincijk)*delz
    GO TO 40

c
c      z-directed total incident current
c
44  rdrhat=(i-.5)*dxcosa+(j-1)*dycosb+(k-0.5)*dzcosg
    tprime=time-rdrhat*cinv-tdelay
    hyincijk=0.0
    IF ((tprime.LT.ttrun).AND.(tprime.GT.0.)) THEN
        INCLUDE 'source.h'
        hyincijk=hamp1y*source
    ENDIF

    rdrhat=(i-1.5)*dxcosa+(j-1)*dycosb+(k-0.5)*dzcosg
    tprime=time-rdrhat*cinv-tdelay
    hyinciljk=0.0
    IF ((tprime.LT.ttrun).AND.(tprime.GT.0.)) THEN
        INCLUDE 'source.h'
        hyinciljk=hamp1y*source
    ENDIF

    rdrhat=(i-1)*dxcosa+(j-1.5)*dycosb+(k-0.5)*dzcosg
    tprime=time-rdrhat*cinv-tdelay
    hxincijlk=0.0

```

```

IF ((tprime.LT.ttrun).AND.(tprime.GT.0.)) THEN
  INCLUDE 'source.h'
  hxincijlk=hamplx*source
ENDIF

rdrhat=(i-1)*dxcosa+(j-.5)*dycosb+(k-0.5)*dzcosg
tprime=time-rdrhat*cinv-tdelay
hxincijk=0.0
IF ((tprime.LT.ttrun).AND.(tprime.GT.0.)) THEN
  INCLUDE 'source.h'
  hxincijk=hamplx*source
ENDIF

field=(hyincijk-hyinciljk)*dely+
$(hxincijlk-hxincijk)*delx

C
40  RETURN
    END

C
    SUBROUTINE SAMTYP(type,typtxt)

C
    INTEGER type,i
    CHARACTER*55 typtxt

C
    This subroutine returns a text string based upon the
C
    sample type defined for a point or slice sensor. The
C
    text string is then written to the diagnostics file.
C
    DO 10 i=1,55
      typtxt(i:i)=' '
10  CONTINUE
      GO TO (11,12,13,14,15,16,17,18,19,20,21,22,23,
$24,25,26,27,28,29,30,31,32,33,34,35,36,37,38,39,41,42,43,44) type

C
    Ex scattered field
C
11  typtxt='Sensor samples scattered x-directed electric field.'
    GO TO 40

C
    Ey scattered field
C
12  typtxt='Sensor samples scattered y-directed electric field.'
    GO TO 40

C
    Ez scattered field
C
13  typtxt='Sensor samples scattered z-directed electric field.'
    GO TO 40

C
    Hx scattered field
C
14  typtxt='Sensor samples scattered x-directed magnetic field.'
    GO TO 40

C
    Hy scattered field
C
15  typtxt='Sensor samples scattered y-directed magnetic field.'
    GO TO 40

```

```

c      Hz scattered field
c
c      16  typtxt='Sensor samples scattered z-directed magnetic field.'
          GO TO 40
c
c      x-directed conduction current
c
c      17  typtxt='Sensor samples x-directed conduction current.'
          GO TO 40
c
c      y-directed conduction current
c
c      18  typtxt='Sensor samples y-directed conduction current.'
          GO TO 40
c
c      z-directed conduction current
c
c      19  typtxt='Sensor samples z-directed conduction current.'
          GO TO 40
c
c      x-directed displacement current
c
c      20  typtxt='Sensor samples x-directed displacement current.'
          GO TO 40
c
c      y-directed displacement current
c
c      21  typtxt='Sensor samples y-directed displacement current.'
          GO TO 40
c
c      z-directed displacement current
c
c      22  typtxt='Sensor samples z-directed displacement current.'
          GO TO 40
c
c      x-directed total current
c
c      23  typtxt='Sensor samples x-directed total current.'
          GO TO 40
c
c      y-directed total current
c
c      24  typtxt='Sensor samples y-directed total current.'
          GO TO 40
c
c      z-directed total current
c
c      25  typtxt='Sensor samples z-directed total current.'
          GO TO 40
c
c      Ex total field
c
c      26  typtxt='Sensor samples total x-directed electric field.'
          GO TO 40
c
c      Ey total field

```

```

C
27  typtxt='Sensor samples total y-directed electric field.'
    GO TO 40
C
C    Ez total field
C
28  typtxt='Sensor samples total z-directed electric field.'
    GO TO 40
C
C    Hx total field
C
29  typtxt='Sensor samples total x-directed magnetic field.'
    GO TO 40
C
C    Hy total field
C
30  typtxt='Sensor samples total y-directed magnetic field.'
    GO TO 40
C
C    Hz total field
C
31  typtxt='Sensor samples total z-directed magnetic field.'
    GO TO 40
C
C    Ex incident field
C
32  typtxt='Sensor samples incident x-directed electric field.'
    GO TO 40
C
C    Ey incident field
C
33  typtxt='Sensor samples incident y-directed electric field.'
    GO TO 40
C
C    Ez incident field
C
34  typtxt='Sensor samples incident z-directed electric field.'
    GO TO 40
C
C    Hx incident field
C
35  typtxt='Sensor samples incident x-directed magnetic field.'
    GO TO 40
C
C    Hy incident field
C
36  typtxt='Sensor samples incident y-directed magnetic field.'
    GO TO 40
C
C    Hz incident field
C
37  typtxt='Sensor samples incident z-directed magnetic field.'
    GO TO 40
C
C    x-directed total scattered current
C
38  typtxt='Sensor samples x-directed total scattered current.'
    GO TO 40

```

```

C      y-directed total scattered current
C
C      39  typtxt='Sensor samples y-directed total scattered current.'
        GO TO 40
C
C      z-directed total scattered current
C
C      41  typtxt='Sensor samples z-directed total scattered current.'
        GO TO 40
C
C      x-directed total incident current
C
C      42  typtxt='Sensor samples x-directed total incident current.'
        GO TO 40
C
C      y-directed total incident current
C
C      43  typtxt='Sensor samples y-directed total incident current.'
        GO TO 40
C
C      z-directed total incident current
C
C      44  typtxt='Sensor samples z-directed total incident current.'
        GO TO 40
40     RETURN
      END

```

c234567

```
C
      SUBROUTINE FINFF
C
      INCLUDE 'main.h'
      REAL uphi,utheta,wphi,wtheta,ephi,etheta,ephiin,ethin,
      $exi,eyi,ezi
      INTEGER m
C
      This subroutine finishes the near-to-far field transformation
      by transforming the U and W vector potentials to far-field
      electric field (Etheta and Ephi) components.
C
      *****
C
      Local variable dictionary
C
      ephi=far field electric field component in phi direction
      ephiin=far field incident electric field in phi direction
      etheta=far field electric field component in theta direction
      ethin=far field incident electric field in theta direction
      exi=incident x-directed electric field at center of space
      eyi=incident y-directed electric field at center of space
      ezi=incident z-directed electric field at center of space
      m=loop counter over time bins of vector potential array
      uphi=far field vector potential U in phi direction
      utheta=far field vector potential U in theta direction
      wphi=far field vector potential W in phi direction
      wtheta=far field vector potential W in theta direction
C
      *****
C
      First compute the direction cosines for far field observation
      point.
C
      cosphi=cos(phi*degrad)
      sinphi=sin(phi*degrad)
      costh=cos(theta*degrad)
      sinth=sin(theta*degrad)
C
      Open the farfield info file
C
      OPEN (22,FILE='FZINFO.DAT')
      REWIND 22
C
      Write header information for fzproc code
C
      WRITE (22,*) delx,dely,delz,delt,tsteps,1
      WRITE (22,*) phi,theta
C
      CLOSE(22)
C
      Compute distance term for center of space
C
      rdrhat=xc*cosa+yc*cosb+zc*cosg
C
      Open the far field data file
C
```

```

OPEN (21,FILE='FZOUT3D.DAT')
REWIND 21

DO 20 m=1,tsteps
C
C   Compute tprime variable for source function
C
tprime=AMAX1(0.0,REAL(m-1)*delt-rf*cinv)
tprime=tprime-rdrhat*cinv-tdelay
C
C   Compute incident electric field terms
C
IF ((tprime.LT.ttrun).AND.(tprime.GT.0.)) THEN
    INCLUDE 'source.h'
    exi=eamplx*source
    eyi=eamply*source
    ezi=eamplz*source
ELSE
    exi=0.0
    eyi=0.0
    ezi=0.0
ENDIF
C
C   Transform incident field to spherical components
C
ethin=(exi*cosphi+eyi*sinphi)*costh-ezi*sinth
ephiin=-exi*sinphi+eyi*cosphi

C
C   write(*,*)'exi=',exi
C   write(*,*)'sinphi=',sinphi
C   write(*,*)'eyi=',eyi
C   write(*,*)'cosphi=',cosphi
C   write(*,*)'ethin=',ethin
C   write(*,*)'ephiin=',ephiin
C
C   Now transform the Cartesian vector potentials into spherical
C   vector potentials according to the Cartesian to spherical
C   vector transformation
C
    utheta=(uandw(1,m)*cosphi+uandw(2,m)*sinphi)*costh-
$uandw(3,m)*sinth
C    write(*,*)'utheta=',utheta
C    write(*,*)'ethin=',ethin
C    write(*,*)'ephiin=',ephiin
C
C   Now transform the Cartesian vector potentials into spherical
C   vector potentials according to the Cartesian to spherical
C   vector transformation
C
    utheta=(uandw(1,m)*cosphi+uandw(2,m)*sinphi)*costh-
$uandw(3,m)*sinth
C    write(*,*)'utheta=',utheta
C    uphi=-uandw(1,m)*sinphi+uandw(2,m)*cosphi
C    write(*,*)'uphi=',uphi
C    wtheta=(uandw(4,m)*cosphi+uandw(5,m)*sinphi)*costh-
$uandw(6,m)*sinth
C    write(*,*)'wtheta=',wtheta
C    wphi=-uandw(4,m)*sinphi+uandw(5,m)*cosphi

```

```

C      Compute far field electric field components
C
C      etheta=-eta0*wtheta-uphi
C      write(*,*)'etheta=',etheta
C      ephi=-eta0*wphi+utheta
C      write(*,*)'ephi=',ephi
C      read(*,*)
C
C      Write data to the far field data file
C
C      WRITE (21,*) m,ephi,etheta,ephiin,ethin
C
20  CONTINUE
    CLOSE (UNIT=21)
    RETURN
    END

```

c234567

c Include file 'farfld.h'

c  
c This file contains all of the variables used for the near  
c to far field transformation.c

c  
c\*\*\*\*\*

c  
c Variable dictionary

c  
c ffc=far field multiplying constant  
c ffield=far field integration surface field components  
c fftime=real time used to compute retarded time  
c fpcdt=constant multiplier =  $1/(4\pi \cdot c \cdot \text{deltat})$   
c ilow=i index of lower integration surface with normal -a\_x  
c inuwx=index for uandw array for integration surfaces with  
c unit normal of +/- a\_x  
c inuwy=index for uandw array for integration surfaces with  
c unit normal of +/- a\_y  
c inuwz=index for uandw array for integration surfaces with  
c unit normal of +/- a\_z  
c iup=i index of upper integration surface with normal +a\_x  
c jlow=j index of lower integration surface with normal -a\_y  
c jup=j index of upper integration surface with normal +a\_y  
c klow=k index of lower integration surface with normal -a\_z  
c kup=k index of upper integration surface with normal +a\_z  
c rf=maximum distance from center of space to a point on the  
c integration surface  
c tretx=retarded time delay for integration surfaces with unit  
c normals in +/- a\_x directions  
c treti=retarded time delay for integration surfaces with unit  
c normals in +/- a\_y directions  
c tretz=retarded time delay for integration surfaces with unit  
c normals in +/- a\_z directions  
c tretdt=retarded time divided by deltat  
c uandw=vector potential array (contains both u and w vector  
c potentials)

c  
c\*\*\*\*\*

REAL tretx,treti,tretz,ftime,ffc,ffield,tretdt,rf,  
\$fpcdt,uandw  
INTEGER inuwx,inuwy,inuwz,ilow,jlow,klow,iup,jup,kup  
COMMON/FAR/tretx(0:ny-9,0:nz-9,0:2),treti(0:nx-9,0:nz-9,0:2),  
\$tretz(0:nx-9,0:ny-9,0:2),uandw(0:6,0:mmax),ftime(0:4),ffc,  
\$ffield(0:4),tretdt,fpcdt,rf  
COMMON/FARSH/inuwx(0:4),inuwy(0:4),inuwz(0:4),ilow,jlow,klow,  
\$iup,jup,kup

c234567

```
C
SUBROUTINE FARFLD

C
INCLUDE 'main.h'
INTEGER i,j,k,iin,jin,kin,inuw
INTEGER m

C
C   This subroutine updates the far field vector
C   potentials U and W.
C
C*****
C
C   Local variable dictionary
C
C   i=cell coordinate number in x direction
C   iin=index into retarded time delay arrays for x
C       direction and also index for field components on
C       integration surfaces with unit normal +/- a_x
C   inuw=index into vector potential array uandw to
C       update the correct vector potential component
C       (ux, uy, uz, wx, wy or wz)
C   j=cell coordinate number in y direction
C   jin=index into retarded time delay arrays for y
C       direction and also index for field components on
C       integration surfaces with unit normal +/- a_y
C   k=cell coordinate number in z direction
C   kin=index into retarded time delay arrays for z
C       direction and also index for field components on
C       integration surfaces with unit normal +/- a_z
C   lfld=loop counter over the four vector potential
C       components to be updated
C   m=time bin index for far field vector potential array
C
C*****
C
C   Set the far field time variables first
C   The first 2 array locations are for magnetic fields
C   so therefore are 1/2 time step off from the electric
C   fields.
C
C   fftime(1)=time-delta2
C   fftime(2)=time-delta2
C   fftime(3)=time-delta
C   fftime(4)=time-delta

C
C   Start integration on faces with unit normals +/- a_x.
C   Start with the faces with unit normal - a_x.
C
C   ffc=dely*delz*fpcdt
C   iin=ilow
C   DO 40 i=1,2
C       DO 30 k=klow,kup
C           kin=k-klow+1
C           DO 20 j=jlow,jup
C               jin=j-jlow+1

C
C       Compute the four field components tangent to the
```

```

C      integration cell. The field component is assumed
C      to be located at the center of the cell.
C
      ffield(1)=0.25*ffc*(hzscat(iin,j,k)+hzscat(iin,j,k+1)+
$hzscat(iin-1,j,k)+hzscat(iin-1,j,k+1))
      ffield(2)=-0.25*ffc*(hyscat(iin,j,k)+hyscat(iin,j+1,k)+
$hyscat(iin-1,j,k)+hyscat(iin-1,j+1,k))
      ffield(3)=-0.5*ffc*(ezscat(iin,j,k)+ezscat(iin,j+1,k))
      ffield(4)=0.5*ffc*(eyscat(iin,j,k)+eyscat(iin,j,k+1))

C
C      Update the corresponding vector potential component
C      and time bin using the first field component.
C
      inuw=inuwx(1)
      tretdt=(fftime(1)+tretx(jin,kin,i))*dtinv
      m=INT(tretdt+0.5)
      uandw(inuw,m-1)=uandw(inuw,m-1)+(0.5-tretdt+m)*
$ffield(1)
      uandw(inuw,m)=uandw(inuw,m)+2.0*(tretdt-m)*
$ffield(1)
      uandw(inuw,m+1)=uandw(inuw,m+1)-(0.5+tretdt-m)*
$ffield(1)

C
C      Update the corresponding vector potential component
C      and time bin using the second field component.
C
      inuw=inuwx(2)
      tretdt=(fftime(2)+tretx(jin,kin,i))*dtinv
      m=INT(tretdt+0.5)
      uandw(inuw,m-1)=uandw(inuw,m-1)+(0.5-tretdt+m)*
$ffield(2)
      uandw(inuw,m)=uandw(inuw,m)+2.0*(tretdt-m)*
$ffield(2)
      uandw(inuw,m+1)=uandw(inuw,m+1)-(0.5+tretdt-m)*
$ffield(2)

C
C      Update the corresponding vector potential component
C      and time bin using the third field component.
C
      inuw=inuwx(3)
      tretdt=(fftime(3)+tretx(jin,kin,i))*dtinv
      m=INT(tretdt+0.5)
      uandw(inuw,m-1)=uandw(inuw,m-1)+(0.5-tretdt+m)*
$ffield(3)
      uandw(inuw,m)=uandw(inuw,m)+2.0*(tretdt-m)*
$ffield(3)
      uandw(inuw,m+1)=uandw(inuw,m+1)-(0.5+tretdt-m)*
$ffield(3)

C
C      Update the corresponding vector potential component
C      and time bin using the fourth field component.
C
      inuw=inuwx(4)
      tretdt=(fftime(4)+tretx(jin,kin,i))*dtinv
      m=INT(tretdt+0.5)
      uandw(inuw,m-1)=uandw(inuw,m-1)+(0.5-tretdt+m)*
$ffield(4)
      uandw(inuw,m)=uandw(inuw,m)+2.0*(tretdt-m)*

```

```

$ffield(4)
      uandw(inuw,m+1)=uandw(inuw,m+1)-(0.5+tretdt-m)*
$ffield(4)
20      CONTINUE
30      CONTINUE
C
C      Now switch to face with unit normal + a_x, so the
C      i index and far field constant must be changed.
C
      ffc=-ffc
      iin=iup
40      CONTINUE
C
C      Next integrate on faces with unit normals +/- a_y.
C      Start with the faces with unit normal - a_y.
C
      ffc=delx*delz*fpcdt
      jin=jlow
      DO 80 j=1,2
        DO 70 k=klow,kup
          kin=k-klow+1
          DO 60 i=ilow,iup
C
C          Compute the four field components tangent to the
C          integration cell. The field component is assumed
C          to be located at the center of the cell.
C
            iin=i-ilow+1
            ffield(1)=-0.25*ffc*(hzscat(i,jin,k)+hzscat(i,jin,k+1)+
$hzscat(i,jin-1,k)+hzscat(i,jin-1,k+1))
            ffield(2)=0.25*ffc*(hxscat(i,jin,k)+hxscat(i+1,jin,k)+
$hxscat(i,jin-1,k)+hxscat(i+1,jin-1,k))
            ffield(3)=0.5*ffc*(ezscat(i,jin,k)+ezscat(i+1,jin,k))
            ffield(4)=-0.5*ffc*(exscat(i,jin,k)+exscat(i,jin,k+1))
C
C          Update the corresponding vector potential component
C          and time bin using the first field component.
C
            inuw=inuw(1)
            tretdt=(fftime(1)+trety(iin,kin,j))*dtinv
            m=INT(tretdt+0.5)
            uandw(inuw,m-1)=uandw(inuw,m-1)+(0.5-tretdt+m)*
$ffield(1)
            uandw(inuw,m)=uandw(inuw,m)+2.0*(tretdt-m)*
$ffield(1)
            uandw(inuw,m+1)=uandw(inuw,m+1)-(0.5+tretdt-m)*
$ffield(1)
C
C          Update the corresponding vector potential component
C          and time bin using the second field component.
C
            inuw=inuw(2)
            tretdt=(fftime(2)+trety(iin,kin,j))*dtinv
            m=INT(tretdt+0.5)
            uandw(inuw,m-1)=uandw(inuw,m-1)+(0.5-tretdt+m)*
$ffield(2)
            uandw(inuw,m)=uandw(inuw,m)+2.0*(tretdt-m)*
$ffield(2)

```

```

        uandw(inuw,m+1)=uandw(inuw,m+1)-(0.5+tretdt-m)*
$ffield(2)
C      Update the corresponding vector potential component
C      and time bin using the third field component.
C
        inuw=inuw(3)
        tretdt=(fftime(3)+tetry(iin,kin,j))*dtinv
        m=INT(tretdt+0.5)
        uandw(inuw,m-1)=uandw(inuw,m-1)+(0.5-tretdt+m)*
$ffield(3)
        uandw(inuw,m)=uandw(inuw,m)+2.0*(tretdt-m)*
$ffield(3)
        uandw(inuw,m+1)=uandw(inuw,m+1)-(0.5+tretdt-m)*
$ffield(3)
C      Update the corresponding vector potential component
C      and time bin using the fourth field component.
C
        inuw=inuw(4)
        tretdt=(fftime(4)+tetry(iin,kin,j))*dtinv
        m=INT(tretdt+0.5)
        uandw(inuw,m-1)=uandw(inuw,m-1)+(0.5-tretdt+m)*
$ffield(4)
        uandw(inuw,m)=uandw(inuw,m)+2.0*(tretdt-m)*
$ffield(4)
        uandw(inuw,m+1)=uandw(inuw,m+1)-(0.5+tretdt-m)*
$ffield(4)
60      CONTINUE
70      CONTINUE
C      Now switch to face with unit normal + a_y, so the
C      j index and far field constant must be changed.
C
        ffc=-ffc
        jin=jup
80      CONTINUE
C      Finally integrate on faces with unit normals +/- a_z.
C      Start with the faces with unit normal - a_z.
C
        ffc=delx*dely*fpcdt
        kin=klow
        DO 120 k=1,2
            DO 110 j=jlow,jup
                jin=j-jlow+1
                DO 100 i=ilow,iup
C
C      Compute the four field components tangent to the
C      integration cell. The field component is assumed
C      to be located at the center of the cell.
C
                iin=i-ilow+1
                ffield(1)=0.25*ffc*(hyscat(i,j,kin)+hyscat(i,j+1,kin)+
$hyscat(i,j,kin-1)+hyscat(i,j+1,kin-1))
                ffield(2)=-0.25*ffc*(hxscat(i,j,kin)+hxscat(i+1,j,kin)+
$hxscat(i,j,kin-1)+hxscat(i+1,j,kin-1))
                ffield(3)=-0.5*ffc*(eyscat(i,j,kin)+eyscat(i+1,j,kin))

```

```

      ffield(4)=0.5*ffc*(exscat(i,j,kin)+exscat(i,j+1,kin))
C
C      Update the corresponding vector potential component
C      and time bin using the first field component.
C
      inuw=inuwz(1)
      tretdt=(fftime(1)+tretz(iin,jin,k))*dtinv
      m=INT(tretdt+0.5)
      uandw(inuw,m-1)=uandw(inuw,m-1)+(0.5-tretdt+m)*
$ffield(1)
      uandw(inuw,m)=uandw(inuw,m)+2.0*(tretdt-m)*
$ffield(1)
      uandw(inuw,m+1)=uandw(inuw,m+1)-(0.5+tretdt-m)*
$ffield(1)

C
C      Update the corresponding vector potential component
C      and time bin using the second field component.
C
      inuw=inuwz(2)
      tretdt=(fftime(2)+tretz(iin,jin,k))*dtinv
      m=INT(tretdt+0.5)
      uandw(inuw,m-1)=uandw(inuw,m-1)+(0.5-tretdt+m)*
$ffield(2)
      uandw(inuw,m)=uandw(inuw,m)+2.0*(tretdt-m)*
$ffield(2)
      uandw(inuw,m+1)=uandw(inuw,m+1)-(0.5+tretdt-m)*
$ffield(2)

C
C      Update the corresponding vector potential component
C      and time bin using the third field component.
C
      inuw=inuwz(3)
      tretdt=(fftime(3)+tretz(iin,jin,k))*dtinv
      m=INT(tretdt+0.5)
      uandw(inuw,m-1)=uandw(inuw,m-1)+(0.5-tretdt+m)*
$ffield(3)
      uandw(inuw,m)=uandw(inuw,m)+2.0*(tretdt-m)*
$ffield(3)
      uandw(inuw,m+1)=uandw(inuw,m+1)-(0.5+tretdt-m)*
$ffield(3)

C
C      Update the corresponding vector potential component
C      and time bin using the fourth field component.
C
      inuw=inuwz(4)
      tretdt=(fftime(4)+tretz(iin,jin,k))*dtinv
      m=INT(tretdt+0.5)
      uandw(inuw,m-1)=uandw(inuw,m-1)+(0.5-tretdt+m)*
$ffield(4)
      uandw(inuw,m)=uandw(inuw,m)+2.0*(tretdt-m)*
$ffield(4)
      uandw(inuw,m+1)=uandw(inuw,m+1)-(0.5+tretdt-m)*
$ffield(4)
100      CONTINUE
110      CONTINUE
      close(10)

C
C      Now switch to face with unit normal + a_z, so the

```

```
      k index and far field constant must be changed.
c
c      ffc=-ffc
      kin=kup
120  CONTINUE
      RETURN
      END
```

c234567

SUBROUTINE ERRCHK

C

INCLUDE 'main.h'  
INTEGER i,j,k,ibeg,jbeg,kbeg,iend,jend,kend

C

C

C

C

C

C

C

C

This subroutine checks for various errors associated with running an FDTD problem.

Check to see if any portion of the object lies outside the far field integration surface.

IF (ffldon) THEN  
  ibeg=nx  
  jbeg=ny  
  kbeg=nz  
  iend=0  
  jend=0  
  kend=0  
  DO 500 k=2,nz1  
    DO 400 j=2,ny1  
      DO 300 i=2,nx1  
        IF ((id1(i,j,k).NE.0).OR.(id2(i,j,k).NE.0).OR.  
\$ (id3(i,j,k).NE.0)) THEN  
          IF (i.LT.ibeg) ibeg=i  
          IF (j.LT.jbeg) jbeg=j  
          IF (k.LT.kbeg) kbeg=k  
          IF (i.GT.iend) iend=i  
          IF (j.GT.jend) jend=j  
          IF (k.GT.kend) kend=k

ENDIF

300 CONTINUE

400 CONTINUE

500 CONTINUE

IF ((iend.GE.iup)) THEN

WRITE (15,\*) '  
WRITE (15,\*) 'Error! The extent of the object is >='  
WRITE (15,\*) 'the upper limit of the far field '  
WRITE (15,\*) 'transformation integration surface in'  
WRITE (15,\*) 'the i direction. Please increase the'  
WRITE (15,\*) 'parameter nx in file setup.h to compensate.'  
WRITE (15,\*) 'The far field integration surface is located'  
WRITE (15,\*) 'at i=',iup,' while the object extends to '  
WRITE (15,\*) 'i=',iend  
WRITE (15,\*) '  
errflg=true

ENDIF

IF ((jend.GE.jup)) THEN

WRITE (15,\*) '  
WRITE (15,\*) 'Error! The extent of the object is >='  
WRITE (15,\*) 'the upper limit of the far field '  
WRITE (15,\*) 'transformation integration surface in'  
WRITE (15,\*) 'the j direction. Please increase the'  
WRITE (15,\*) 'parameter ny in file setup.h to compensate.'  
WRITE (15,\*) 'The far field integration surface is located'  
WRITE (15,\*) 'at j=',jup,' while the object extends to '  
WRITE (15,\*) 'j=',jend

```

WRITE (15,*) ' '
errflg=true
ENDIF
IF ((kend.GE.kup)) THEN
WRITE (15,*) ' '
WRITE (15,*) 'Error! The extent of the object is >='
WRITE (15,*) 'the upper limit of the far field '
WRITE (15,*) 'transformation integration surface in'
WRITE (15,*) 'the k direction. Please increase the'
WRITE (15,*) 'parameter nz in file setup.h to compensate.'
WRITE (15,*) 'The far field integration surface is located'
WRITE (15,*) 'at k=',kup,' while the object extends to '
WRITE (15,*) 'k=',kend
WRITE (15,*) ' '
errflg=true
ENDIF
IF ((ibeg.LE.ilow)) THEN
WRITE (15,*) ' '
WRITE (15,*) 'Error! The extent of the object is <='
WRITE (15,*) 'the lower limit of the far field '
WRITE (15,*) 'transformation integration surface in'
WRITE (15,*) 'the i direction. Please increase the'
WRITE (15,*) 'parameter nx in file setup.h to compensate.'
WRITE (15,*) 'The far field integration surface is located'
WRITE (15,*) 'at i=',ilow,' while the object extends to '
WRITE (15,*) 'i=',ibeg
WRITE (15,*) ' '
errflg=true
ENDIF
IF ((jbeg.LE.jlow)) THEN
WRITE (15,*) ' '
WRITE (15,*) 'Error! The extent of the object is <='
WRITE (15,*) 'the lower limit of the far field '
WRITE (15,*) 'transformation integration surface in'
WRITE (15,*) 'the j direction. Please increase the'
WRITE (15,*) 'parameter ny in file setup.h to compensate.'
WRITE (15,*) 'The far field integration surface is located'
WRITE (15,*) 'at j=',jlow,' while the object extends to '
WRITE (15,*) 'j=',jbeg
WRITE (15,*) ' '
errflg=true
ENDIF
IF ((kbeg.LE.klow)) THEN
WRITE (15,*) ' '
WRITE (15,*) 'Error! The extent of the object is <='
WRITE (15,*) 'the lower limit of the far field '
WRITE (15,*) 'transformation integration surface in'
WRITE (15,*) 'the k direction. Please increase the'
WRITE (15,*) 'parameter nz in file setup.h to compensate.'
WRITE (15,*) 'The far field integration surface is located'
WRITE (15,*) 'at k=',klow,' while the object extends to '
WRITE (15,*) 'k=',kbeg
WRITE (15,*) ' '
errflg=true
ENDIF
ENDIF
ENDIF

```

c  
c Check the point source feed location

C

```

IF (pntsrc) THEN
  IF ((iptsrc.LE.1).OR.(iptsrc.GE.nx)) THEN
    WRITE (15,*) ' '
    WRITE (15,*) 'Error! The i location for the point'
    WRITE (15,*) 'source is incorrectly specified. Its'
    WRITE (15,*) 'current value is iptsrc=',iptsrc
    WRITE (15,*) 'Please correct the value in either'
    WRITE (15,*) 'the file defaults.f or userdefs.f.'
    WRITE (15,*) ' '
    errflg=true
  ENDIF
  IF ((jptsrc.LE.1).OR.(jptsrc.GE.ny)) THEN
    WRITE (15,*) ' '
    WRITE (15,*) 'Error! The j location for the point'
    WRITE (15,*) 'source is incorrectly specified. Its'
    WRITE (15,*) 'current value is jptsrc=',jptsrc
    WRITE (15,*) 'Please correct the value in either'
    WRITE (15,*) 'the file defaults.f or userdefs.f.'
    WRITE (15,*) ' '
    errflg=true
  ENDIF
  IF ((kptsrc.LE.1).OR.(kptsrc.GE.nz)) THEN
    WRITE (15,*) ' '
    WRITE (15,*) 'Error! The k location for the point'
    WRITE (15,*) 'source is incorrectly specified. Its'
    WRITE (15,*) 'current value is kptsrc=',kptsrc
    WRITE (15,*) 'Please correct the value in either'
    WRITE (15,*) 'the file defaults.f or userdefs.f.'
    WRITE (15,*) ' '
    errflg=true
  ENDIF
  IF ((fdtype.NE.'x').OR.(fdtype.NE.'y').OR.
$(fdtype.NE.'z')) THEN
    WRITE (15,*) ' '
    WRITE (15,*) 'Error! The feed type for the point'
    WRITE (15,*) 'source is incorrectly specified. Its'
    WRITE (15,*) 'current value is fdtype=',fdtype
    WRITE (15,*) 'Please correct the value in either'
    WRITE (15,*) 'the file defaults.f or userdefs.f.'
    WRITE (15,*) ' '
    errflg=true
  ENDIF
ENDIF
IF (errflg) THEN
  WRITE (15,*) 'Execution halted.'
  CLOSE (UNIT=15)
  STOP
ELSE
  WRITE (15,*) 'No errors reported.'
  CLOSE (UNIT=15)
ENDIF
RETURN
END

```

c234567

```
c
c      SUBROUTINE UPDEXS
c
c      This subroutine updates the x component of scattered
c      electric field.
c
c      INCLUDE 'main.h'
c      INTEGER i,j,k
c
c*****
c
c      Local variable dictionary
c
c      i=cell coordinate number in x direction
c      j=cell coordinate number in y direction
c      k=cell coordinate number in z direction
c*****
c
c      DO 30 k=2,nz1
c        DO 20 j=2,ny1
c          DO 10 i=1,nx1
c
c            Get the material type
c
c            if(id1(i,j,k).eq.0) then
c
c              Free space update equation
c
c              exscat(i,j,k)=exscat(i,j,k)+(hzscat(i,j,k)-
c$hzscat(i,j-1,k))*dtoedy-
c$(hyscat(i,j,k)-hyscat(i,j,k-1))*dtoedz
c
c            elseif(id1(i,j,k).eq.1) then
c
c              Perfect conductor
c
c              exscat(i,j,k)=0.0
c              IF (.NOT.plwave) GO TO 10
c              rdrhat=(i-0.5)*dxcosa+(j-1)*dycosb+(k-1)*dzcosg
c              tprime=time-rdrhat*cinv-tdelay
c              IF ((tprime.LT.ttrun).AND.(tprime.GT.0.)) THEN
c                INCLUDE 'source.h'
c                exscat(i,j,k)=-eamplx*source
c              ENDIF
c
c            else
c
c              Lossy dielectric materials
c
c              exscat(i,j,k)=exscat(i,j,k)*eold(id1(i,j,k))+
c$(hzscat(i,j,k)-hzscat(i,j-1,k))*dhdy(id1(i,j,k))-
c$(hyscat(i,j,k)-hyscat(i,j,k-1))*dhdz(id1(i,j,k))
c              IF (.NOT.plwave) GO TO 10
c              rdrhat=(i-0.5)*dxcosa+(j-1)*dycosb+(k-1)*dzcosg
c              tprime=time-rdrhat*cinv-tdelay
c              IF ((tprime.LT.ttrun).AND.(tprime.GT.0.)) THEN
```

```

        INCLUDE 'source.h'
        INCLUDE 'ddtsrce.h'
        exscat(i,j,k)=exscat(i,j,k)-eamplx*
$(einc(id1(i,j,k))*source+ddtein(id1(i,j,k))*ddtsrc)
        ENDIF
C
        endif
10      CONTINUE
20      CONTINUE
30      CONTINUE
      RETURN
      END
C
      SUBROUTINE UPDEYS
C
C      This subroutine updates the y component of scattered
C      electric field.
C
      INCLUDE 'main.h'
      INTEGER i,j,k
C
C*****
C
C      Local variable dictionary
C
C      i=cell coordinate number in x direction
C      j=cell coordinate number in y direction
C      k=cell coordinate number in z direction
C*****
C
      DO 30 k=2,nz1
        DO 20 j=1,ny1
          DO 10 i=2,nx1
C
C          Get the material type
C
          if (id2(i,j,k).EQ.0) then
C
C          Free space update equation
C
          ey scat(i,j,k)=ey scat(i,j,k)-(hz scat(i,j,k)-
$hz scat(i-1,j,k))*dtoedx+
$(hx scat(i,j,k)-hx scat(i,j,k-1))*dtoedz
C
          elseif (id2(i,j,k).eq.1) then
C
C          Perfect conductor
C
          ey scat(i,j,k)=0.0
          IF (.NOT.plwave) GO TO 10
          rdrhat=(i-1)*dxcosa+(j-0.5)*dycosb+(k-1)*dzcosg
          tprime=time-rdrhat*cinv-tdelay
          IF ((tprime.LT.ttrun).AND.(tprime.GT.0.)) THEN
            INCLUDE 'source.h'
            ey scat(i,j,k)=-eamply*source
          ENDIF
C

```

```

else
C      Lossy dielectric materials
C
C      ey scat(i,j,k)=ey scat(i,j,k)*eold(id2(i,j,k))-
$(hzscat(i,j,k)-hzscat(i-1,j,k))*dhdx(id2(i,j,k))+
$(hxscat(i,j,k)-hxscat(i,j,k-1))*dhdz(id2(i,j,k))
      IF (.NOT.plwave) GO TO 10
      rdrhat=(i-1)*dxcosa+(j-0.5)*dycosb+(k-1)*dzcosg
      tprime=time-rdrhat*cinv-tdelay
      IF ((tprime.LT.ttrun).AND.(tprime.GT.0.)) THEN
          INCLUDE 'source.h'
          INCLUDE 'ddtsrce.h'
          ey scat(i,j,k)=ey scat(i,j,k)-eamply*
          $(einc(id2(i,j,k))*source+ddtein(id2(i,j,k))*ddtsrc)
      ENDIF
C
C      endif
10      CONTINUE
20      CONTINUE
30      CONTINUE
      RETURN
      END
C
      SUBROUTINE UPDEZS
C
C      This subroutine updates the z component of scattered
C      electric field.
C
C      INCLUDE 'main.h'
C      INTEGER i,j,k
C
C*****
C
C      Local variable dictionary
C
C      i=cell coordinate number in x direction
C      j=cell coordinate number in y direction
C      k=cell coordinate number in z direction
C*****
C
C      DO 30 k=1,nz1
C        DO 20 j=2,ny1
C          DO 10 i=2,nx1
C
C            Get the material type
C
C            if (id3(i,j,k).eq.0) then
C
C              Free space update equation
C
C              ezscat(i,j,k)=ezscat(i,j,k)-(hxscat(i,j,k)-
                $hxscat(i,j-1,k))*dtoedy+
                $(hyscat(i,j,k)-hyscat(i-1,j,k))*dtoedx
C
C              elseif (id3(i,j,k).eq.1) then

```

```

c          Perfect conductor
c
      ezscat(i,j,k)=0.0
      IF (.NOT.plwave) GO TO 10
      rdrhat=(i-1)*dxcosa+(j-1)*dycosb+(k-0.5)*dzcosg
      tprime=time-rdrhat*cinv-tdelay
      IF ((tprime.LT.ttrun).AND.(tprime.GT.0.)) THEN
        INCLUDE 'source.h'
        ezscat(i,j,k)=-eamplz*source
      ENDIF

c
      else

c          Lossy dielectric materials
c
c      ezscat(i,j,k)=ezscat(i,j,k)*eold(id3(i,j,k))-
      $(hxscat(i,j,k)-hxscat(i,j-1,k))*dhdy(id3(i,j,k))+
      $(hyscat(i,j,k)-hyscat(i-1,j,k))*dhdx(id3(i,j,k))
      IF (.NOT.plwave) GO TO 10
      rdrhat=(i-1)*dxcosa+(j-1)*dycosb+(k-0.5)*dzcosg
      tprime=time-rdrhat*cinv-tdelay
      IF ((tprime.LT.ttrun).AND.(tprime.GT.0.)) THEN
        INCLUDE 'source.h'
        INCLUDE 'ddtsrce.h'
        ezscat(i,j,k)=ezscat(i,j,k)-eamplz*
        $(einc(id3(i,j,k))*source+ddteinc(id3(i,j,k))*ddtsrc)
      ENDIF

c
      endif
10      CONTINUE
20      CONTINUE
30      CONTINUE
      RETURN
      END

```

```

C      SUBROUTINE DEFLT5
C      INCLUDE 'main.h'
C      INTEGER m

C      This subroutine sets up the defaults for source pulse function
C      and all of the necessary parameters for its specification.
C      *****
C      Local variable dictionary
C      m=loop counter
C      *****
C      Gaussian pulse
C      IF (gauss) THEN
C        IF (tspec) THEN
C          Time specifications
C          IF (rise) THEN
C            trise=20.0*delt
C          ELSE
C            pwidth=51.0*delt
C            ap=0.5
C          ENDIF
C        ELSEIF (fspec) THEN
C          Frequency specification
C          fup=c/(10.0*AMAX1(delx,dely,delz))
C          adb=80.0
C        ENDIF
C      ELSEIF (banlim) THEN
C        Bandlimited pulse
C        IF (tspec) THEN
C          Time specifications
C          IF (rise) THEN
C            trise=20.0*delt
C          ELSE
C            pwidth=51.0*delt
C            ap=0.5
C          ENDIF
C        ELSEIF (fspec) THEN
C          Frequency specification
C          flow=0.0
C          bw=2.0*c/(10.0*AMAX1(delx,dely,delz))
C          adb=140.0
C        ENDIF

```

```

ELSEIF (hypsec) THEN
C
C      Hyperbolic secant pulse
C
      IF (tspec) THEN
C
C          Time specifications
C
          ah=0.5
          ah=1.0/EXP(1.0)
          pwidth=51.0*delt
      ELSEIF (fspec) THEN
C
          ah=1.0/EXP(1.0)
          pwidth=51.0*delt
C
          WRITE (15,*) ' '
          WRITE (15,*) 'Error! The flag fspec is set to true for the'
          WRITE (15,*) 'hyperbolic secant pulse. This pulse can only'
          WRITE (15,*) 'be specified in the time domain and the flag'
          WRITE (15,*) 'tspec must be set to true.'
          WRITE (15,*) 'Make sure all parameters'
          WRITE (15,*) 'for the hyperbolic secant pulse are defined.'
          errflg=true
      ENDIF
ELSEIF (rsine) THEN
C
C      Ramped sinusoid function
C
          f0=1.0/(20.0*delt)
          cycles=10
          ttrun=tsteps*delt
      ELSEIF (step) THEN
C
C          Unit step function
C
          IF (tspec) THEN
              IF (rise) THEN
                  trise=40.0E-12
              ELSE
                  trise=40.0E-12
                  WRITE (15,*) ' '
                  WRITE (15,*) 'Error! The flag rise for specifying the'
                  WRITE (15,*) 'unit step function is set to false. Please'
                  WRITE (15,*) 'set this flag to true in file setup.h.'
                  WRITE (15,*) 'Dont forget to set the rise time for the '
                  WRITE (15,*) 'step function in either defaults.f or '
                  WRITE (15,*) 'userdefs.f.'
                  errflg=true
              ENDIF
          ELSE
              trise=40.0E-12
              WRITE (15,*) ' '
              WRITE (15,*) 'Error! The flag fspec is set to true for the'
              WRITE (15,*) 'unit step pulse. This pulse can only'
              WRITE (15,*) 'be specified in the time domain and the flag'
              WRITE (15,*) 'tspec must be set to true.'
              WRITE (15,*) 'Make sure the rise time for the unit step'

```

```

        WRITE (15,*) 'pulse is defined in file defaults.f or file '
        WRITE (15,*) 'userdefs.f.'
        errflg=true
    ENDIF
ELSE
C
    WRITE (15,*) ' '
    WRITE (15,*) 'Error! All flags for choosing an incident'
    WRITE (15,*) 'source function are set to false. Please choose'
    WRITE (15,*) 'the appropriate source function by setting its'
    WRITE (15,*) 'flag to "true" and by setting the appropriate'
    WRITE (15,*) 'time or frequency specification flags (tspec or'
    WRITE (15,*) 'fspec) and by setting the appropriate '
    WRITE (15,*) 'parameters in file setup.h.'
    errflg=true
ENDIF
C
C Initialize material parameter defaults
C
DO 10 m=1,maxmat
    eps(m)=eps0
    mu(m)=mu0
    sigma(m)=0.0
    msigma(m)=0.0
10 CONTINUE
C
C Initialize sensor defaults
C
CALL DFSENS
C
C Initialize point source defaults
C
    iptsrc=nx/2
    jptsrc=ny/2
    kptsrc=nz/2
    fdtype='y'
C
RETURN
END

```

c234567

```
c      This file contains all of the time derivatives of the
c      source functions for the source pulse type.
c
c      IF (gauss) THEN
c
c          Time derivative of Gaussian pulse
c
c          ddtsrc=-2.0*(tprime-toff)*tau0i*tau0i*source
c
c      ELSEIF (banlim) THEN
c
c          Time derivative of bandlimited pulse
c
c          ddtsrc=EXP(-((tprime-toff)*tau0i)**2)*
c          $(w0*COS(w0*(tprime-toff))-2.0*(tprime-toff)*tau0i*tau0i*
c          $SIN(w0*(tprime-toff)))
c
c      ELSEIF (hypsec) THEN
c
c          Time derivative of hyperbolic secant pulse
c
c          ddtsrc=-2.0*tau0i*(EXP((tprime-toff)*tau0i)-
c          $EXP(-(tprime-toff)*tau0i))/((EXP(-(tprime-toff)*tau0i)+
c          $EXP((tprime-toff)*tau0i)))
c
c      ELSEIF (rsine) THEN
c
c          Time derivative of ramped sinusoid
c
c          ddtsrc=-w0*SIN(w0*(tprime-toff))+
c          $EXP(-((tprime-toff)*tau0i)**2)*
c          $(w0+2.0*(tprime-toff)*tau0i*tau0i*COS(w0*(tprime-toff)))
c
c      ENDIF
```

```

c      constants.h
c234567
c      This file defines constants that are set once within
c      the code and never change.  Examples are C (speed of light),
c      PI, etc.  Do not put variables in this file!
c
c*****
c
c      Variable dictionary
c
c      c=speed of light in vacuum
c      cinv=1/c
c      degrad=degrees to radians conversion factor (= pi/180)
c      e=2.71828
c      false=logical variable (= .FALSE.)
c      eps0=permittivity of free space
c      eta0=free space wave impedance
c      mu0=permeability of free space
c      pi=3.14
c      twopi=2.0*pi
c      true=logical variable (= .TRUE.)
c
c*****
c
c      REAL c,pi,eps0,mu0,eta0,twopi,cinv,degrad,e
c      COMMON/CONSTA/c,pi,eps0,mu0,eta0,twopi,cinv,degrad,e
c      LOGICAL*1 true,false
c      PARAMETER (true=.TRUE.,false=.FALSE.)

```

```

C
C PROGRAM FZPROC*****edit
C subroutine fzproctemsub
C
C This program computes backscatter vs. frequency radar
C cross-section from data files generated by the Penn State
C University Finite Difference Time Domain 3D computer codes.
C If desired, this program can also extract the far-zone
C scatterd fields vs. time and the incident field vs. time.
C This code accompanies the 3D codes fdtcd and fdtdd.
C The program computes sigma (rcs) in dBsm. The dependence on
C distance r has been suppressed.
C
C Revised 15 NOV 1993
C
C *****
C
C MAIN PROGRAM
C
C *****
C
C These parameters set the size of the arrays for this program.
C MAXFFT is the total number of FFT2 samples to be computed.
C MAXFFT must be a power of 2.
C NFZ is the maximum number of far-zone angles that can be
C computed.
C
C PARAMETER (MAXFFT=65536,NFZ=32)
C
C Array declarations follow
C
C REAL THETFZ(NFZ), PHIFZ(NFZ)
C REAL EPHIRE(1,MAXFFT), EPHII(1,MAXFFT)
C REAL ETHRE(1,MAXFFT), ETHI(1,MAXFFT)
C REAL PHIRE(NFZ,MAXFFT), PHIIM(NFZ,MAXFFT)
C REAL THRE(NFZ,MAXFFT), THIM(NFZ,MAXFFT)
C REAL DELX, DELY, DELZ, DT, DELF
C INTEGER NSTOP, NUMFZ, NFMAX
C
C Read in the far-zone time domain information.
C
C CALL READIN(DELX,DELY,DELZ,DT,NSTOP,NUMFZ,
1 PHIFZ,THETFZ,EPHIRE,EPHII,ETHRE,ETHI,
2 PHIRE,PHIIM,THRE,THIM,MAXFFT,NFZ)
C
C Write out the time domain data to some files
C
C CALL WRTTIM(NSTOP,NUMFZ,DT,PHIFZ,THETFZ,EPHIRE,
1 ETHRE,PHIRE,THRE,MAXFFT,NFZ)
C
C Take FFT2s of the data
C
C CALL CMPFFT(DELX,DELY,DELZ,DT,NSTOP,NUMFZ,
1 PHIFZ,THETFZ,EPHIRE,EPHII,ETHRE,ETHI,
2 PHIRE,PHIIM,THRE,THIM,MAXFFT,NFZ,DELF,NFMAX)
C
C Compute the RCS
C

```

```

      CALL CMPCRS (NUMFZ,NFMAX,DELF,
1  EPHIRE,EPHII,ETHRE,ETHI,PHIRE,PHIIM,THRE,THIM,
2  MAXFFT,NFZ)
C
C  Write the RCS data a file
C
      CALL WRTRCS (NUMFZ,NFMAX,DELF,THETFZ,PHIFZ,
1  EPHIRE,EPHII,ETHRE,ETHI,PHIRE,PHIIM,THRE,THIM,
2  MAXFFT,NFZ)
C
C  Finished
C
C  STOP*****edit
C  RETURN
C  END
C *****
      SUBROUTINE READIN (DELX,DELY,DELZ,DT,NSTOP,NUMFZ,
1  PHIFZ,THETFZ,EPHIRE,EPHII,ETHRE,ETHI,
2  PHIRE,PHIIM,THRE,THIM,MAXFFT,NFZ)

      REAL THETFZ (NFZ), PHIFZ (NFZ)
      REAL EPHIRE (1,MAXFFT), EPHII (1,MAXFFT)
      REAL ETHRE (1,MAXFFT), ETHI (1,MAXFFT)
      REAL PHIRE (NFZ,MAXFFT), PHIIM (NFZ,MAXFFT)
      REAL THRE (NFZ,MAXFFT), THIM (NFZ,MAXFFT)
      REAL DELX, DELY, DELZ, DT
      INTEGER NSTOP, NUMFZ

C
C  This subroutine reads in the far-zone time domain data
C  from the files fzinfo.dat and fzout3d.dat or fzout3d.bin
C  depending on how many far-zone angles are specified.
C
C  open the far zone information file
C
C  OPEN (UNIT=25,FILE='farfld.dat',STATUS='OLD')*****edit
C  OPEN (UNIT=25,FILE='FZINFO.DAT',STATUS='OLD')
C
C  read in header information
C
C  READ (25,*) DELX,DELY,DELZ,DT,NSTOP,NUMFZ
C
C  Error check on NUMFZ to make sure it is not larger
C  than values declared for this program.
C
      IBAD = 0
      IF (NUMFZ.GT.NFZ) THEN
        WRITE (*,*) 'Increase parameter NFZ to at least ', NUMFZ
        IBAD = 1
      ENDIF
C
C  compute amount of zero padding that can be added to
C  the input data. If NPAD equals zero, then the input
C  file is longer than the number of FFT2 samples, so
C  MAXFFT will have to be increased to at least NSTOP.
C
      NPAD = MAXFFT/NSTOP
      IF (NPAD.EQ.0) THEN
        WRITE (*,*) 'The number of FFT samples is less than the length'

```

```

WRITE(*,*) 'of the input data. Increase the parameter MAXFFT'
WRITE(*,*) 'to at least ', NSTOP
IBAD = 1
ENDIF

C
C If any errors were encountered, stop the program
C
IF (IBAD.EQ.1) STOP

C
C Open the ASCII far zone data file
C This file contains far-zone time-domain fields for the
C first far-zone angle.
C
OPEN (UNIT=30,FILE='fzout3d.dat',STATUS='OLD')*****edit
OPEN (UNIT=30,FILE='FZOUT3D.DAT',STATUS='OLD')

C
C if there is more than one far-zone angle, then
C open the binary far-zone time-domain data file
C to read in all the far-zone data.
C
IF (NUMFZ.GT.1) THEN
  OPEN (UNIT=35,FILE='fzout3d.bin',STATUS='OLD',
1    FORM='UNFORMATTED')
ENDIF

C
C Read in the data
C
DO 10 L=1,NUMFZ

C
C read in far-zone scattering angles from fzinfo.dat
C
  READ (25,*) PHIFZ(L),THETFZ(L)*****edit (this was commented out)
  READ (25,*) PHIFZ(L),THETFZ(L)
10 CONTINUE
  CLOSE (UNIT=25)

C
C read the time domain far-zone field values from the
C far-zone file (fzout3d.dat or fzout3d.bin)
C
DO 30 I=1,NSTOP

C
C set imaginary parts of fields to zero
C
  EPHII(1,I)=0.0
  ETHI(1,I)=0.0
  PHIIM(1,I)=0.0
  THIM(1,I)=0.0

C
C read in scattered and incident fields in binary form for
C multiple angles (NUMFZ > 1) or ASCII form for single angle
C (NUMFZ = 1)
C
  READ (25,*) L,PHIRE(1,I),THRE(1,I),EPHIRE(1,I),ETHRE(1,I)*****edit
  READ (30,*) L,PHIRE(1,I),THRE(1,I),EPHIRE(1,I),ETHRE(1,I)
  IF (NUMFZ.GT.1) THEN
    READ (35) (PHIRE(L,I),THRE(L,I),L=1,NUMFZ)
    DO 20, L = 1,NUMFZ
      PHIIM(L,I)=0.0

```

```

      THIM(L,I)=0.0
20      CONTINUE
      ENDIF
30      CONTINUE
      CLOSE (UNIT=30)
      IF (NUMFZ.GT.1) CLOSE (UNIT=35)
      RETURN
      END
C *****
      SUBROUTINE WRTTIM(NSTOP,NUMFZ,DT,PHIFZ,THETFZ,EPHIRE,
1 ETHRE,PHIRE,THRE,MAXFFT,NFZ)
C
      REAL THETFZ(NFZ), PHIFZ(NFZ)
      REAL EPHIRE(1,MAXFFT), ETHRE(1,MAXFFT)
      REAL PHIRE(NFZ,MAXFFT), THRE(NFZ,MAXFFT)
      REAL DT
      INTEGER NSTOP, NUMFZ
      CHARACTER*1 ANS
C
C      This subroutine writes the far-zone field data vs. time
C      into a file.  If only one far-zone angle exists, then this
C      procedure will be automatic, otherwise the angles available
C      will be displayed and user input will determine what
C      fields will be plotted.
C
C      If there is only one far-zone angle...
C
      IF (NUMFZ.EQ.1) THEN
        WRITE(*,*) ' '
        WRITE(*,*) 'Writing time-domain far-zone scattered',
1 ' fields to file fzscat.dat'
        WRITE(*,*) ' '
        WRITE(*,*) 'Writing time-domain far-zone incident',
1 ' fields to file fzinc.dat'
        WRITE(*,*) ' '
        OPEN(UNIT=40,FILE='fzscat.dat',STATUS='UNKNOWN')
        OPEN(UNIT=41,FILE='fzinc.dat',STATUS='UNKNOWN')
        WRITE(40,100)
        WRITE(40,200)
        WRITE(40,300)
        WRITE(41,400)
        WRITE(41,200)
        WRITE(41,300)
        DO 10, I=1,NSTOP
          WRITE(40,*) (DT*I)*1.e9, PHIRE(1,I), THRE(1,I)
          WRITE(41,*) (DT*I)*1.e9, EPHIRE(1,I), ETHRE(1,I)
10      CONTINUE
          CLOSE(40)
          CLOSE(41)
        ELSE
C
C      Otherwise, there are several angles to choose from so
C      this procedure can not be automatic.  Ask if time
C      domain data is desired and if so, call a separate
C      subroutine to deal with it.
C
        WRITE(*,*) ' '

```



```

WRITE(*,*) 'Number      Phi      Theta'
WRITE(*,*) '-----'
K=0
DO 10, I=1,NUMFZ
  K = K + 1
  WRITE(*,500) I, PHIFZ(I), THETFZ(I)
  IF(K.EQ.10) THEN
    WRITE(*,*) 'Press ENTER to see more angles'
    READ(5,1000) CERR
    K = 0
  ENDIF
10 CONTINUE
  WRITE(*,501) NUMFZ+1
  WRITE(*,*) 'Choose one of the numbers above, 0 to quit'
  WRITE(*,*) 'Enter your choice -> '
  READ(*,*) IANG
C
C   If a non-existence choice was made, re-ask the question.
C
C       IF((IANG.LT.0).OR.(IANG.GT.(NUMFZ+1))) GO TO 5
C
C   If a single angle is to be plotted, save it in a file
C
C       IF((IANG.NE.0).AND.(IANG.NE.(NUMFZ+1))) THEN
C         IF(IANG.LT.10) THEN
C           C1 = ITOC(IANG)
C           FLNAM = 'fzsc'//C1//'.dat'
C         ELSE
C           C2 = ITOC(IANG)
C           FLNAM = 'fzsc'//C2//'.dat'
C         ENDIF
C         OPEN(UNIT=40,FILE=FLNAM,STATUS='UNKNOWN')
C         WRITE(40,100)
C         WRITE(40,101)
C         WRITE(40,102) PHIFZ(IANG), THETFZ(IANG)
C         WRITE(40,101)
C         WRITE(40,200)
C         WRITE(40,300)
C         DO 20, I=1,NSTOP
C           WRITE(40,*) (DT*I)*1.e9, PHIRE(IANG,I), THRE(IANG,I)
20      CONTINUE
        CLOSE(40)
        WRITE(*,*) ' Writing time domain far-zone field to file ',FLNAM
        ELSE IF(IANG.EQ.(NUMFZ+1)) THEN
C
C   If choice was to print out all of the time-domain fields
C   into files...
C
C       DO 30, J=1,NUMFZ
C         IF(J.LT.10) THEN
C           C1 = ITOC(J)
C           FLNAM = 'fzsc'//C1//'.dat'
C         ELSE
C           C2 = ITOC(J)
C           FLNAM = 'fzsc'//C2//'.dat'
C         ENDIF
C         OPEN(UNIT=40,FILE=FLNAM,STATUS='UNKNOWN')
C         WRITE(40,100)

```

```

        WRITE(40,101)
        WRITE(40,102) PHIFZ(J), THETFZ(J)
        WRITE(40,101)
        WRITE(40,200)
        WRITE(40,300)
        DO 40, I=1,NSTOP
            WRITE(40,*) (DT*I)*1.e9, PHIRE(J,I), THRE(J,I)
40      CONTINUE
        CLOSE(40)
        WRITE(*,*) ' Writing time domain far-zone field to file ',
1          FLNAM
30      CONTINUE
    ENDIF
C      If choice was not 'quit' or 'all of the above', return to
C      top of menu.
C      IF((IANG.NE.0).AND.(IANG.NE.(NUMFZ+1))) GO TO 5
C
C      Write out the far-zone incident field
C
C      WRITE(*,*) 'Writing time-domain far-zone incident',
1      ' fields to file fzinc.dat'
        OPEN(UNIT=41,FILE='fzinc.dat',STATUS='UNKNOWN')
        WRITE(41,400)
        WRITE(41,200)
        WRITE(41,300)
        DO 50, I=1,NSTOP
            WRITE(41,*) (DT*I)*1.e9, EPHIRE(1,I), ETHRE(1,I)
50      CONTINUE
C
C      Finished
C
C      100 FORMAT ('# Far-zone scattered field (volts/m) vs. Time')
101 FORMAT ('#')
102 FORMAT ('# Phi = ', F6.2, ' Theta = ', F6.2)
200 FORMAT ('# Time (ns) Ephi (V/m) Etheta (V/m)')
300 FORMAT ('#-----')
400 FORMAT ('# Far-zone incident field (volts/m) vs. Time')
500 FORMAT (2X,I2,5X,F7.2, 5X, F7.2)
501 FORMAT (2X,I2,' All of the above')
1000 FORMAT (80A1)
        RETURN
        END
C *****
SUBROUTINE CMPFFT(DELX,DELY,DELZ,DT,NSTOP,NUMFZ,
1 PHIFZ,THETFZ,EPHIRE,EPHII,ETHRE,ETHI,
2 PHIRE,PHIIM,THRE,THIM,MAXFFT,NFZ,DELF,NFMAX)
C
    REAL THETFZ(NFZ), PHIFZ(NFZ)
    REAL EPHIRE(1,MAXFFT), EPHII(1,MAXFFT)
    REAL ETHRE(1,MAXFFT), ETHI(1,MAXFFT)
    REAL PHIRE(NFZ,MAXFFT), PHIIM(NFZ,MAXFFT)
    REAL THRE(NFZ,MAXFFT), THIM(NFZ,MAXFFT)
    REAL DELX, DELY, DELZ, DT, DELF
    INTEGER NSTOP, NUMFZ, NFMAX
C
C

```



```

          THIM(L,I)=0.0
93      CONTINUE
          C
          compute Fourier transform of scattered field
C
C      IF (NUMFZ.GT.1) WRITE(*,600) L
          CALL FFT2(PHIRE,PHIIM,NMAX,NU,L,NFZ)
          CALL FFT2(THRE,THIM,NMAX,NU,L,NFZ)
94      CONTINUE
          C
          WRITE(*,*) '...Computing FFT of incident fields...'
C
C      compute Fourier transform of incident fields
C
          CALL FFT2(EPHIRE,EPHII,NMAX,NU,1,1)
          CALL FFT2(ETHRE,ETHI,NMAX,NU,1,1)
C
C      Finished
C
600     FORMAT(' ...Computing Far-Zone Angle ', I3, '...')
          RETURN
          END
C *****
          SUBROUTINE CMPRCS(NUMFZ,NFMAX,DELF,
1      EPHIRE,EPHII,ETHRE,ETHI,PHIRE,PHIIM,THRE,THIM,
2      MAXFFT,NFZ)
C
          COMPLEX J,ARG,CINC
          REAL EPHIRE(1,MAXFFT), EPHII(1,MAXFFT)
          REAL ETHRE(1,MAXFFT), ETHI(1,MAXFFT)
          REAL PHIRE(NFZ,MAXFFT), PHIIM(NFZ,MAXFFT)
          REAL THRE(NFZ,MAXFFT), THIM(NFZ,MAXFFT)
          REAL DELF
          REAL MAGPS, MAGTS, MAGI
          INTEGER NUMFZ, NFMAX
C
C      This subroutine computes the RCS
C
C
          PI=4.0*ATAN(1.0)
          J=CMPLX(0.0,1.0)
C
          WRITE(*,*) '...Computing RCS...'
C
          DO 160 L=1,NUMFZ
              DO 150 I=1,NFMAX
C
C          Compute magnitude of phi and theta-pol scattered field
C          and complex incident field.
C
          MAGPS=SQRT(PHIRE(L,I)**2+PHIIM(L,I)**2)
          MAGTS=SQRT(THRE(L,I)**2+THIM(L,I)**2)
          CINC=CSQRT((EPHIRE(1,I)+J*EPHII(1,I))**2+
1      (ETHRE(1,I)+J*ETHI(1,I))**2)
          MAGI=CABS(CINC)
C
C      Compute phase of RCS

```

```

C      ARG=(PHIRE(L,I)+J*PHIIM(L,I))/CINC
      PHIIM(L,I)=180./PI*ATAN2C(AIMAG(ARG),REAL(ARG))
      ARG=(THRE(L,I)+J*THIM(L,I))/CINC
      THIM(L,I)=180./PI*ATAN2C(AIMAG(ARG),REAL(ARG))

C      Compute magnitude of RCS

C      PHIRE(L,I)=10.*LOG10(4.0*PI*(MAGPS/MAGI)**2)
      THRE(L,I)=10.*LOG10(4.0*PI*(MAGTS/MAGI)**2)

150    CONTINUE
160    CONTINUE

C      Finished

C      RETURN
      END
C *****
      SUBROUTINE WRTRCS(NUMFZ,NFMAX,DELF,THETFZ,PHIFZ,
1  EPHIRE,EPHII,ETHRE,ETHI,PHIRE,PHIIM,THRE,THIM,
2  MAXFFT,NFZ)

C      REAL THETFZ(NFZ), PHIFZ(NFZ)
      REAL EPHIRE(1,MAXFFT), EPHII(1,MAXFFT)
      REAL ETHRE(1,MAXFFT), ETHI(1,MAXFFT)
      REAL PHIRE(NFZ,MAXFFT), PHIIM(NFZ,MAXFFT)
      REAL THRE(NFZ,MAXFFT), THIM(NFZ,MAXFFT)
      REAL DELF, MAGPS, MAGTS
      INTEGER NUMFZ, NFMAX
      COMPLEX CJ, CINC, ARG

C      This subroutine writes the RCS output to files.
C      If only one far-zone angle exists, then this is
C      automatic. Otherwise, some input is needed from
C      the user to determine what type of output is desired.
C      Available choices are:
C      Backscatter RCS vs. Frequency -> single far-zone angle
C
C      Backscatter and Bistatic RCS vs. Frequency -> multiple
C      far-zone angles
C      Bistatic RCS vs. Angle -> multiple far-zone angles
C
C      CJ = CMPLX(0.0,1.0)
      PI = 2.0*ACOS(0.0)
      IF (NUMFZ.EQ.1) THEN

C      Since there's only one far-zone angle, there is no need to
C      ask for any input.

C      WRITE(*,*) ' '
      WRITE(*,*) 'Writing RCS vs. Frequency to file 3dracs.dat'
      WRITE(*,*) ' '
      WRITE(*,*) 'Writing incident pulse spectrum to file 3dinc.dat'
      WRITE(*,*) ' '
      OPEN(UNIT=40,FILE='3dracs.dat',STATUS='UNKNOWN')
      OPEN(UNIT=42,FILE='3dracs.PHIRE.dat',STATUS='UNKNOWN')
      OPEN(UNIT=43,FILE='3dracs.THRE.dat',STATUS='UNKNOWN')
      OPEN(UNIT=41,FILE='3dinc.dat',STATUS='UNKNOWN')

```

```

WRITE(40,100)
WRITE(40,101)
WRITE(40,200)
WRITE(40,201)
WRITE(40,300)
WRITE(41,400)
WRITE(41,101)
WRITE(41,200)
WRITE(41,201)
WRITE(41,300)
DO 10, I=1,NFMAX
  WRITE(40,500) I*DELF*1.e-9, PHIRE(1,I), THRE(1,I),
    PHIIM(1,I),THIM(1,I)
  WRITE(42,500) I*DELF*1.e-9, PHIRE(1,I)
  WRITE(43,500) I*DELF*1.e-9, THRE(1,I)
C
C
C
  Convert the incident field spectrum to magnitude and phase
  CINC=EPHIRE(1,I)+J*EPHII(1,I)
  MAGPS=SQRT(EPHIRE(1,I)**2+EPHII(1,I)**2)
  ARG=CINC
  PHAPS=180./PI*ATAN2C(AIMAG(ARG),REAL(ARG))
  IF (MAGPS.GT.0.0) MAGPS=20.*LOG10(MAGPS)
  CINC=ETHRE(1,I)+J*ETHI(1,I)
  MAGTS=SQRT(ETHRE(1,I)**2+ETHI(1,I)**2)
  ARG=CINC
  PHATS=180./PI*ATAN2C(AIMAG(ARG),REAL(ARG))
  IF (MAGTS.GT.0.0) MAGTS=20.*LOG10(MAGTS)
  WRITE(41,500) I*DELF*1.e-9, MAGPS, MAGTS, PHAPS, PHATS
10 CONTINUE
  CLOSE(40)
  CLOSE(41)
  ELSE
C
C
C
  If there are multiple far-zone angles, there are choices for
  what type of output is to be plotted. So, call a separate
  subroutine to perform the input/output for this case.
C
  CALL RCSMRE(NUMFZ,NFMAX,DELF,THETFZ,PHIFZ,
1    EPHIRE,EPHII,ETHRE,ETHI,PHIRE,PHIIM,THRE,THIM,
2    MAXFFT,NFZ)
  ENDIF
C
C
C
  Finished
C
100 FORMAT('# Radar Cross-Section vs. Frequency')
101 FORMAT('#')
200 FORMAT('# Frequency          Magnitude (dBsm)          Phase',
1 '(degrees)')
201 FORMAT('# (GHz)          EphI          Etheta          EphI ',
1 'Etheta')
300 FORMAT('#-----',
1 '-----')
400 FORMAT('# Incident Pulse Spectrum')
500 FORMAT(2X,F10.7,3X,F10.5,1X,F10.5,1X,F10.5,1X,F10.5)
  RETURN
  END
C *****

```



```

C
C      Convert the incident field spectrum to magnitude and phase
C
      CINC=EPHIRE(1,I)+J*EPHII(1,I)
      MAGPS=SQRT(EPHIRE(1,I)**2+EPHII(1,I)**2)
      ARG=CINC
      PHAPS=180./PI*ATAN2C(AIMAG(ARG),REAL(ARG))
      IF (MAGPS.GT.0.0) MAGPS=20.*LOG10(MAGPS)
      CINC=ETHRE(1,I)+J*ETHI(1,I)
      MAGTS=SQRT(ETHRE(1,I)**2+ETHI(1,I)**2)
      ARG=CINC
      PHATS=180./PI*ATAN2C(AIMAG(ARG),REAL(ARG))
      IF (MAGTS.GT.0.0) MAGTS=20.*LOG10(MAGTS)
      WRITE(41,400) I*DELF*1.e-9, MAGPS, MAGTS, PHAPS, PHATS
10  CONTINUE
      CLOSE(41)
C
C      Finished
C
      100 FORMAT('# Incident Pulse Spectrum')
101  FORMAT('#')
200  FORMAT('# Frequency          Magnitude (dBsm)          Phase',
           1 ' (degrees)')
201  FORMAT('# (GHz)          Ephi          Etheta          Ephi ',
           1 '          Etheta')
300  FORMAT('# -----',
           1 '-----')
400  FORMAT(2X,F10.7,3X,F10.5,1X,F10.5,1X,F10.5,1X,F10.5)
      RETURN
      END
C *****
      SUBROUTINE RCSVF(NUMFZ,NFMAX,DELF,THETFZ,PHIFZ,
1  EPHIRE,EPHII,ETHRE,ETHI,PHIRE,PHIIM,THRE,THIM,
2  MAXFFT,NFZ)
C
      REAL THETFZ(NFZ), PHIFZ(NFZ)
      REAL EPHIRE(1,MAXFFT), EPHII(1,MAXFFT)
      REAL ETHRE(1,MAXFFT), ETHI(1,MAXFFT)
      REAL PHIRE(NFZ,MAXFFT), PHIIM(NFZ,MAXFFT)
      REAL THRE(NFZ,MAXFFT), THIM(NFZ,MAXFFT)
      REAL DELF
      INTEGER NUMFZ, NFMAX
      COMPLEX CINC, ARG
      CHARACTER*25 FLNAM
      CHARACTER*1 C1, CERR
      CHARACTER*2 C2
      CHARACTER*3 ITOC
C
C      This subroutine writes the RCS vs. frequency at one
C      angle.
C
      WRITE(*,*) ' '
      WRITE(*,*) ' '
      WRITE(*,*) ' '
      WRITE(*,*) ' '
      WRITE(*,*) ' '
      WRITE(*,*) ' '
      WRITE(*,*) ' '

```

```

WRITE(*,*) ' '
WRITE(*,*) ' '
WRITE(*,*) ' '
5 WRITE(*,*) ' '
WRITE(*,*) 'The following far-zone angles are available',
1 ' for computing'
WRITE(*,*) 'RCS vs. frequency.'
WRITE(*,*) 'Number      Phi      Theta'
WRITE(*,*) '-----'
K=0
DO 10, I=1, NUMFZ
    K = K + 1
    WRITE(*,502) I, PHIFZ(I), THETFZ(I)
    IF (K.EQ.10) THEN
        WRITE(*,*) 'Press ENTER to see more angles'
        READ(5,1000) CERR
        K = 0
    ENDIF
10 CONTINUE
WRITE(*,501) NUMFZ+1
WRITE(*,*) 'Choose one of the numbers above, 0 to quit'
WRITE(*,*) 'Enter your choice -> '
READ(*,*) IANG
C
C If a non-existence choice was made, re-ask the question.
C
C IF ((IANG.LT.0).OR.(IANG.GT.(NUMFZ+1))) GO TO 5
C
C If a single angle is to be plotted, save it in a file
C
C IF ((IANG.NE.0).AND.(IANG.NE.(NUMFZ+1))) THEN
    IF (IANG.LT.10) THEN
        C1 = ITOC(IANG)
        FLNAM = '3drcs'//C1//'.dat'
    ELSE
        C2 = ITOC(IANG)
        FLNAM = '3drcs'//C2//'.dat'
    ENDIF
    OPEN(UNIT=40,FILE=FLNAM,STATUS='UNKNOWN')
    WRITE(40,100)
    WRITE(40,101)
    WRITE(40,102) PHIFZ(IANG), THETFZ(IANG)
    WRITE(40,101)
    WRITE(40,200)
    WRITE(40,201)
    WRITE(40,300)
    DO 20, I=1, NFMAX
        WRITE(40,400) I*DELF*1.e-9, PHIRE(IANG,I), THRE(IANG,I),
1          PHIIM(IANG,I), THIM(IANG,I)
20 CONTINUE
    CLOSE(40)
    WRITE(*,*) 'Writing RCS vs. Frequency to file ',FLNAM
    ELSE IF (IANG.EQ.(NUMFZ+1)) THEN
        C
        C If choice was to print out the RCS for all the angles
        C into files...
        C
        DO 30, J=1, NUMFZ

```

```

        IF (J.LT.10) THEN
            C1 = ITOC(J)
            FLNAM = '3dracs'//C1//'.dat'
        ELSE
            C2 = ITOC(J)
            FLNAM = '3dracs'//C2//'.dat'
        ENDIF
        OPEN(UNIT=40,FILE=FLNAM,STATUS='UNKNOWN')
        WRITE(40,100)
        WRITE(40,101)
        WRITE(40,102) PHIFZ(J), THETFZ(J)
        WRITE(40,101)
        WRITE(40,200)
        WRITE(40,201)
        WRITE(40,300)
        DO 40, I=1,NFMAX
            WRITE(40,400) I*DELF*1.e-9, PHIRE(J,I), THRE(J,I),
                PHIIM(J,I),THIM(J,I)
1
40    CONTINUE
        CLOSE(40)
        WRITE(*,*) 'Writing RCS vs. Frequency to file ',FLNAM
30    CONTINUE
    ENDIF

C
C    If choice was not 'quit' or 'all of the above', return to
C    top of menu.
C
    IF((IANG.NE.0).AND.(IANG.NE.(NUMFZ+1))) GO TO 5
C
100   FORMAT ('# Radar Cross-Section vs. Frequency')
    101   FORMAT ('#')
102   FORMAT ('# Phi = ', F6.2, ' Theta = ', F6.2)
200   FORMAT ('# Frequency          Magnitude (dBsm)          Phase',
1      ' (degrees)')
201   FORMAT ('# (GHz)          EphI          Etheta          EphI ',
1      ' Etheta')
300   FORMAT ('#-----',
1      '-----')
400   FORMAT (2X,F10.7,3X,F10.5,1X,F10.5,1X,F10.5,1X,F10.5)
    501   FORMAT (2X,I2,' All of the above')
502   FORMAT (2X,I2,5X,F7.2, 5X, F7.2)
1000  FORMAT (80A1)
C
C    Finished
C
    RETURN
    END
C *****
    SUBROUTINE RCSVA (NUMFZ,NFMAX,DELF,THETFZ,PHIFZ,
1    EPHIRE,EPHII,ETHRE,ETHI,PHIRE,PHIIM,THRE,THIM,
2    MAXFFT,NFZ)
C
    REAL THETFZ(NFZ), PHIFZ(NFZ)
    REAL EPHIRE(1,MAXFFT), EPHII(1,MAXFFT)
    REAL ETHRE(1,MAXFFT), ETHI(1,MAXFFT)
    REAL PHIRE(NFZ,MAXFFT), PHIIM(NFZ,MAXFFT)
    REAL THRE(NFZ,MAXFFT), THIM(NFZ,MAXFFT)

```

```

REAL DELF
INTEGER NUMFZ, NFMAX
COMPLEX CINC, ARG
CHARACTER*25 FLNAM
CHARACTER*1 C1, CERR
CHARACTER*2 C2
CHARACTER*3 C3, ITOC
INTEGER IVAL(100)

C
C This subroutine plots the RCS vs. angle at a single
C frequency.
C
FMAX = DELF*NFMAX*1.e-9

C
WRITE(*,*) ' '
WRITE(*,*) ' '
WRITE(*,*) ' '
WRITE(*,*) ' '
WRITE(*,*) ' '
WRITE(*,*) ' '
WRITE(*,*) ' '
WRITE(*,*) ' '
WRITE(*,*) ' '
WRITE(*,*) ' '
WRITE(*,*) ' '
WRITE(*,*) ' '
WRITE(*,*) ' '
WRITE(*,*) 'It is HIGHLY recommended that a steady-state FDTD',
1 ' code be'
WRITE(*,*) 'used for this type of output.'
5 WRITE(*,*) ' '
WRITE(*,*) 'The Bistatic Radar Cross-Section vs. Angle may',
1 ' be plotted'
WRITE(*,*) 'at frequencies below ', FMAX, ' GHz.'
WRITE(*,*) ' '
WRITE(*,*) 'The following far-zone angles are available',
1 ' for computing'
WRITE(*,*) 'RCS vs. Angle.'
WRITE(*,*) 'Number      Phi      Theta'
WRITE(*,*) '-----'
K=0
DO 9, I=1,NUMFZ
      K = K + 1
      WRITE(*,502) I, PHIFZ(I), THETFZ(I)
      IF(K.EQ.10) THEN
        WRITE(*,*) 'Press ENTER to see more angles'
        READ(5,1000) CERR
        K = 0
      ENDIF
9 CONTINUE
WRITE(*,*) ' '
WRITE(*,*) 'Choose a plot type from the following list. '
WRITE(*,*) ' 1. Phi varies, Theta constant '
WRITE(*,*) ' 2. Theta varies, Phi constant '
WRITE(*,*) ' 3. Far-zone angle specified in FDTD code varies'
WRITE(*,*) ' '
WRITE(*,*) 'Enter a choice, 0 to quit '
WRITE(*,*) ' '
READ(*,*) IANS

```

C

```

IF(IANS .EQ. 1) THEN
C
C   If plot is to be versus phi, get a theta angle that will
C   be constant.

WRITE(*,*) ' '
WRITE(*,*) 'Enter the constant value for Theta: '
READ(*,*) ITHETA
NUMPHI = 0
DO 10 I=1,NUMFZ
    IF (INT(THETFZ(I)).EQ.ITHETA) THEN
        RTHETA = THETFZ(I)
        NUMPHI = NUMPHI+1
        IVAL(NUMPHI) = I
    ENDIF
10 CONTINUE
IF (NUMPHI.EQ.0) THEN
    WRITE(*,*) 'No far-zone points were saved in the Theta = ',
1      ITHETA, ' plane'
    WRITE(*,*) 'Press ENTER to continue'
    READ(5,1000) CERR
    GO TO 5
ELSE
    WRITE(*,*) ' '
    WRITE(*,*) 'Enter the frequency for the plot (in GHz): '
    READ(*,*) FREQ
    IF (FREQ.GT.FMAX) THEN
        WRITE(*,*) ' '
        WRITE(*,*) 'Frequency is too high. Enter a value less',
1      ' than', FMAX
        WRITE(*,*) 'Press ENTER to continue'
        READ(5,1000) CERR
        GO TO 5
    ENDIF
    IFREQ = NINT(FREQ*1.e9/DELF) + 1
    IF (IFREQ.LT.10) THEN
        C1 = ITOC(IFREQ)
        FLNAM = 'rcsf'//C1//'.dat'
    ELSE IF (IFREQ.LT.100) THEN
        C2 = ITOC(IFREQ)
        FLNAM = 'rcsf'//C2//'.dat'
    ELSE IF (IFREQ.LT.1000) THEN
        C3 = ITOC(IFREQ)
        FLNAM = 'rcsf'//C3//'.dat'
    ENDIF
    OPEN(UNIT=40,FILE=FLNAM,STATUS='UNKNOWN')
    WRITE(40,100)
    WRITE(40,101)
    WRITE(40,102) FREQ
    WRITE(40,103) IFREQ*DELF
    WRITE(40,104) RTHETA
    WRITE(40,101)
    WRITE(40,200)
    WRITE(40,201)
    WRITE(40,300)
    DO 20 I=1,NUMPHI
        WRITE(40,400) PHIFZ(IVAL(I)),PHIRE(IVAL(I),IFREQ),
1      THRE(IVAL(I),IFREQ),PHIIM(IVAL(I),IFREQ),

```

```

2          THIM(IVAL(I), IFREQ)
20  CONTINUE
    CLOSE(40)
    WRITE(*,*) 'Writing RCS vs. Angle to file ', FLNAM
    ENDIF
ELSE IF (IANS.EQ.2) THEN
    C
C    If plot is to be versus theta, get a phi angle that will
C    be constant.
C
    WRITE(*,*) ' '
    WRITE(*,*) 'Enter the constant value for Phi: '
    READ(*,*) IPHI
    NUMTHET = 0
    DO 11 I=1, NUMFZ
        IF (INT(PHIFZ(I)).EQ.IPHI) THEN
            RPHI = PHIFZ(I)
            NUMTHET = NUMTHET+1
            IVAL(NUMTHET) = I
        ENDIF
    11 CONTINUE
    IF (NUMTHET.EQ.0) THEN
        WRITE(*,*) 'No far-zone points were saved in the Phi = ',
1        IPHI, ' plane'
        WRITE(*,*) 'Press ENTER to continue'
        READ(5,1000) CERR
        GO TO 5
    ELSE
        WRITE(*,*) ' '
        WRITE(*,*) 'Enter the frequency for the plot (in GHz): '
        READ(*,*) FREQ
        IF (FREQ.GT.FMAX) THEN
            WRITE(*,*) ' '
            WRITE(*,*) 'Frequency is too high. Enter a value less',
1            ' than', FMAX
            WRITE(*,*) 'Press ENTER to continue'
            READ(5,1000) CERR
            GO TO 5
        ENDIF
        IFREQ = NINT(FREQ*1.e9/DELF) + 1
        IF (IFREQ.LT.10) THEN
            C1 = ITOC(IFREQ)
            FLNAM = 'rcsf'//C1//'.dat'
        ELSE IF (IFREQ.LT.100) THEN
            C2 = ITOC(IFREQ)
            FLNAM = 'rcsf'//C2//'.dat'
        ELSE IF (IFREQ.LT.1000) THEN
            C3 = ITOC(IFREQ)
            FLNAM = 'rcsf'//C3//'.dat'
        ENDIF
        OPEN(UNIT=40, FILE=FLNAM, STATUS='UNKNOWN')
        WRITE(40,100)
        WRITE(40,101)
        WRITE(40,102) FREQ
        WRITE(40,103) IFREQ*DELF
        WRITE(40,105) RPHI
        WRITE(40,101)
        WRITE(40,202)
    
```

```

        WRITE(40,201)
        WRITE(40,300)
        DO 30 I=1,NUMTHET
            WRITE(40,400) THETFZ(IVAL(I)),PHIRE(IVAL(I),IFREQ),
1                THRE(IVAL(I),IFREQ),PHIIM(IVAL(I),IFREQ),
2                THIM(IVAL(I),IFREQ)
30    CONTINUE
        CLOSE(40)
        WRITE(*,*) 'Writing RCS vs. Angle to file ',FLNAM
        ENDIF
    ELSE IF( IANS.EQ.3) THEN
C
C      Plot versus far-zone angles
C
        WRITE(*,*) ' '
        WRITE(*,*) 'Enter the frequency for the plot (in GHz): '
            READ(*,*) FREQ
        IF(FREQ.GT.FMAX) THEN
            WRITE(*,*) ' '
            WRITE(*,*) 'Frequency is too high. Enter a value less',
1            ' than', FMAX
            WRITE(*,*) 'Press ENTER to continue'
                READ(5,1000) CERR
            GO TO 5
        ENDIF
            IFREQ = NINT(FREQ*1.e9/DELF) + 1
        IF(IFREQ.LT.10) THEN
            C1 = ITOC(IFREQ)
            FLNAM = 'rcsf'//C1//'.dat'
        ELSE IF(IFREQ.LT.100) THEN
            C2 = ITOC(IFREQ)
            FLNAM = 'rcsf'//C2//'.dat'
        ELSE IF(IFREQ.LT.1000) THEN
            C3 = ITOC(IFREQ)
            FLNAM = 'rcsf'//C3//'.dat'
        ENDIF
        OPEN(UNIT=40,FILE=FLNAM,STATUS='UNKNOWN')
        WRITE(40,100)
        WRITE(40,101)
            WRITE(40,102) FREQ
        WRITE(40,103) IFREQ*DELF
        WRITE(40,101)
        WRITE(40,203)
        WRITE(40,201)
        WRITE(40,300)
        DO 31 I=1,NUMFZ
            WRITE(40,401) PHIFZ(I),THETFZ(I),PHIRE(I,IFREQ),
1                THRE(I,IFREQ),PHIIM(I,IFREQ),
2                THIM(I,IFREQ)
31    CONTINUE
        CLOSE(40)
        WRITE(*,*) 'Writing RCS vs. Angle to file ',FLNAM
    ELSE IF( IANS.NE.0) THEN
        GO TO 5
    ENDIF
C
C
C      Finished

```

```

C
100 FORMAT('# Radar Cross-Section versus Angle')
101 FORMAT('#')
102 FORMAT('# Chosen Frequency = ',F6.3,' GHz')
103 FORMAT('# FFT Bin (Actual) Frequency = ',F6.3,' GHz')
104 FORMAT('# Theta Observation Angle = ',F6.2,' degrees')
105 FORMAT('# Phi Observation Angle = ',F6.2,' degrees')
200 FORMAT('# Phi Magnitude (dBsm) Phase',
1 ' (degrees)')
201 FORMAT('# (degrees) Ephi Etheta Ephi ',
1 ' Etheta')
202 FORMAT('# Theta Magnitude (dBsm) Phase',
1 ' (degrees)')
203 FORMAT('# Phi Theta Magnitude (dBsm) Phase',
1 ' (degrees)')
300 FORMAT('#-----',
1 '-----')
400 FORMAT(2X,F10.6,3X,F10.5,1X,F10.5,1X,F10.5,1X,F10.5)
401 FORMAT(2F7.2,3X,F10.5,1X,F10.5,1X,F10.5,1X,F10.5)
502 FORMAT (2X,I2,5X,F7.2, 5X, F7.2)
1000 FORMAT(80A1)
RETURN
END
C *****
C
C FAST FOURIER TRANSFORM SUBROUTINE--THIS SUBROUTINE PERFORMS A
C FOURIER TRANSFORM ON THE COMPLEX INPUT SEQUENCE X AND OVERWRITES
C X ON OUTPUT. THIS SUBROUTINE CAN ALSO BE USED TO COMPUTE THE
C INVERSE FOURIER TRANSFORM BY PERFORMING THE FOLLOWING STEPS:
C
C DO 10 I=1,N
C XIMAG(I)=-XIMAG(I)
C 10 CONTINUE
C
C CALL FFT2(XREAL,XIMAG,N,NU)
C
C DO 20 I=1,N
C XIMAG(I)=-XIMAG(I)/N
C XREAL(I)=XREAL(I)/N
C 20 CONTINUE
C
C N IS THE NUMBER OF SAMPLES (I.E. LENGTH OF SEQUENCE) AND
C NU IS DEFINED AS: N=2**NU.
C
C SUBROUTINE FFT2(XREAL,XIMAG,N,NU,LL,LMAX)
C DIMENSION XREAL(LMAX,N),XIMAG(LMAX,N)
C N2=N/2
C NU1=NU-1
C K=0
C DO 110 L=1,NU
100 DO 105 I=1,N2
P=IBITR2(K/(2**NU1),NU)
ARG=6.283185*P/FLOAT(N)
C=COS(ARG)
S=SIN(ARG)
K1=K+1
K1N2=K1+N2
TREAL=XREAL(LL,K1N2)*C+XIMAG(LL,K1N2)*S

```

```

        TIMAG=XIMAG(LL,K1N2)*C-XREAL(LL,K1N2)*S
        XREAL(LL,K1N2)=XREAL(LL,K1)-TREAL
        XIMAG(LL,K1N2)=XIMAG(LL,K1)-TIMAG
        XREAL(LL,K1)=XREAL(LL,K1)+TREAL
        XIMAG(LL,K1)=XIMAG(LL,K1)+TIMAG
        K=K+1
105     CONTINUE
106     K=K+N2
        IF (K.LT.N) GO TO 100
        K=0
        NU1=NU1-1
        N2=N2/2
110     CONTINUE
        DO 120 K=1,N
            I=IBITR2(K-1,NU)+1
            IF (I.LE.K) GO TO 120
            TREAL=XREAL(LL,K)
            TIMAG=XIMAG(LL,K)
            XREAL(LL,K)=XREAL(LL,I)
            XIMAG(LL,K)=XIMAG(LL,I)
            XREAL(LL,I)=TREAL
            XIMAG(LL,I)=TIMAG
120     CONTINUE
130     RETURN
        END
C *****
C
C     FUNCTION SUBPROGRAM IBITR2(J,NU) FOR USE WITH FFT SUBROUTINE
C     THIS IS A BIT REVERSAL SUBPROGRAM
C
        FUNCTION IBITR2(J,NU)
        J1=J
        IBITR2=0
        DO 200 I=1,NU
            J2=J1/2
            IBITR2=IBITR2*2+(J1-2*J2)
            J1=J2
200     CONTINUE
        RETURN
        END

C *****
C     FUNCTION ATAN2C(A,B)
C
C     This function computes the arctangent function and error
C     checks for situations where both arguments are zero.
C
        IF ((A.EQ.0.0).AND.(B.EQ.0.0)) THEN
            ATAN2C = 0.0
        ELSE
            ATAN2C = ATAN2(A,B)
        ENDIF
        RETURN
        END

C *****
C     CHARACTER*3 FUNCTION ITOC(I)
C
C     Changes I to ascii character for I.  I should be 0-9!

```

```

C
  NHUN=I/100
  NTEN=(I - NHUN*100)/10
  NONE=(I - NHUN*100-NTEN*10)
  IF (NHUN.GT.0) THEN
    ITOC=CHAR(48+NHUN)//CHAR(48+NTEN)//CHAR(48+NONE)
  ELSEIF (NTEN.GT.0) THEN
    ITOC=CHAR(48+NTEN)//CHAR(48+NONE)
  ELSE
    ITOC=CHAR(48+NONE)
  ENDIF
  RETURN
  END
C *****

```

## DISTRIBUTION LIST

AUL/LSE Bldg 1405 - 600 Chennault Circle Maxwell AFB, AL 36112-6424	1 cy
DTIC/OCP 8527 John J. Kingman Rd, Suite 0944 Ft Belvoir, VA 22060-6218	2 cys
AFSAA/SAI 1580 Air Force Pentagon Washington, DC 20330-1580	1 cy
PL/SUL Kirtland AFB, NM 87117-5776	2 cys
PL/HO Kirtland AFB, NM 87117-5776	1 cy
Official Record Copy PL/WSQW/Robert Torres	5 cys